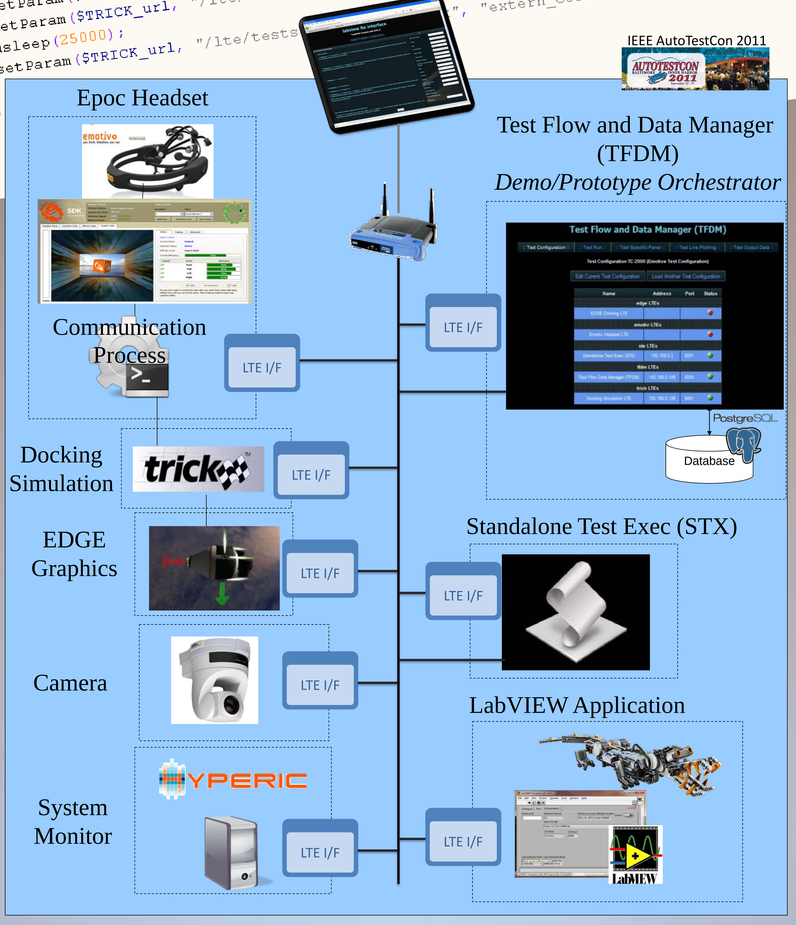
Automation Hooks

Software Orchestration Portable Demonstration



Accelerate… Discovery!

NASA is demonstrating the benefits of an open-standards interface for test orchestration. In this display,

- Data interfaces are being dynamically positioned and then discovered by plug-and-play
- Database tables are being created by a script operating on metadata provided by the data sources
- Test operations are being orchestrated by REST architecture, without a command set
- Assorted operating systems are represented
- Test articles include simulations, hardware, even people, in a distributable configuration
- Web-based scripts are operating on IEEE 1671 ATML-formatted parameters
- A web browser can probe a test orchestration interface– while a test is running

Future work should easily achieve features like **Save**, **Restore**, **DiffReport**, and **DiffView**; a system "**health**" roll-up; and issue tracking. Automation Hooks Architecture seeks to use a spiral approach to converge open standards for tools interfaces to streamline the engineering workflow.

Human Cognitive Technology

Human Cognitive Technology may someday help reduce size, weight, and power of spacecraft controls and provide a third hand for astronauts or ground controllers. Commercially emerging technology could translate brain neural, and electrical signals from muscular responses into commands which can actuate mechanisms using non-invasive, non-gel brain-computer interfaces (BCI). In the long run, neural and muscular-response technologies will allow the operation of crew/spacecraft systems to become a natural extension of the human mind and body for a more organic and intuitive approach to spacecraft operation and control.

Automation Hooks Architecture for Flexible Test Orchestration – Concept Development and Validation

Chatwin A. Lansdowne, John R. Maclean, *Members, IEEE*
Christopher E. Winton, Patrick A. McCartney

***Abstract*— The Automation Hooks Architecture Trade Study for Flexible Test Orchestration sought a standardized data-driven alternative to conventional automated test programming interfaces. The study recommended composing the interface using multicast DNS (mDNS/SD) service discovery, Representational State Transfer (Restful) Web Services, and Automatic Test Markup Language (ATML).**

We describe additional efforts to rapidly mature the Automation Hooks Architecture candidate interface definition by validating it in a broad spectrum of applications. These activities have allowed us to further refine our concepts and provide observations directed toward objectives of economy, scalability, versatility, performance, severability, maintainability, scriptability and others.

***Index Terms*— Software standards, Test equipment, Test facilities, Testing, Software management, Software reusability**

I. Introduction

NASA proposed a foundation for a new open-standards based test orchestration software architecture [1]. The Automation Hooks Architecture is being developed to fulfill a game changing technology need for a simple scalable systems engineering solution which can minimize the largely unspoken lifecycle business costs of performing traditional test control and measurement operations. The intent of the architecture is to achieve these operating cost reductions by providing a non-proprietary framework for improvement and standardization of software automation tools to assist or replace current engineering and science workflows. Increased efficiency is achieved by reducing manual data collection, manual intervention in cycle test procedures and configuration checkpoints and restores, eliminating data format changes between tools, and reducing other labor-intensive non-skilled tasks. The architecture also provides a framework for cumulative knowledge capture which transcribes institutional operational knowledge into explicit instructions and associated documentation.

Manuscript received June 1, 2011. This work was performed in NASA Johnson Space Center's Avionic Systems Division.

C. A. Lansdowne is with the National Aeronautics and Space Administration, Houston, TX 77058 USA (phone: 281-483-1265; fax: 281-483-6297; e-mail: chatwin.lansdowne@ nasa.gov).

J. R. Maclean is with METECS, Houston, TX 77058 USA (phone 281-483-3265; e-mail: john.r.maclean@ nasa.gov).

While addressing cost of operations, the architecture also addresses the increasing embedded complexity of avionic subsystems which require us to "use a computer to test a computer" to provide synchronization, hard stare, and management of detailed configuration and status data that are not practical in manual operations. Machines are simply more attentive and impartial observers than people, and can write faster too.

At the heart of the current architecture is a loosely-coupled highly modular software interface built on platform-independent open standards using open-source implementations widely available from active user communities. A shallow connection to existing software applications was achieved that is inexpensive to integrate and maintain, connecting through a variety of already available Application Program Interfaces (APIs), with data-driven harvest at the origin using a single portable Automation Hooks Architecture (AHA) protocol-interface development.

A resource based web services protocol and widely supported and standardized service-discovery techniques create a machine-discoverable and machine-readable test set interface that can coexist with a user interface; dedicated user interfaces don't scale well, and we believe this interface can. The interface definition is inherently already compatible with a broad assortment of web-based software. Using Representation State Transfer (Rest) software architecture principles (including self-described messages and hypermedia-assisted state transitions) promotes loose coupling, consistency, and transparency. The interface can be self-contained, packed with documentation so that a script author or a machine or a data post-analyst need not look elsewhere. The robust interface stands alone with no middle-ware dependencies and minimal reliance on supporting infrastructure. The interface is intended to require no maintenance of its components or the platform. The Automatic Test Markup Language (ATML) provides a standard set of language constructs for describing test-specific information that integrates nicely into the web-services based interface architecture. The underlying protocol set is very mature and we believe converges API trends that we see in aerospace, test, DoD, and consumer products communities.

As we demonstrate in Section III, this non-proprietary interface is highly versatile, a criterion for broad usage and acceptance.

II. Bestiary

Each independently controlled or monitored module of test equipment or test software is combined with a common AHA interface component to create a Logical Test Element (LTE). We distinguish (Figure 1) between the LTE interface, which implements the AHA protocol, and the LTE application, which controls the hardware or implements the simulation. The LTE interface and the LTE application may be developed by different skill-sets. The interface between the two is referred to as the backend interface. The backend interface will be application specific and several implementations that cover a wide range of NASA requirements have been developed for the examples discussed in this paper. The backend interface is deliberately kept quite shallow to minimize the burden of providing and maintaining it.

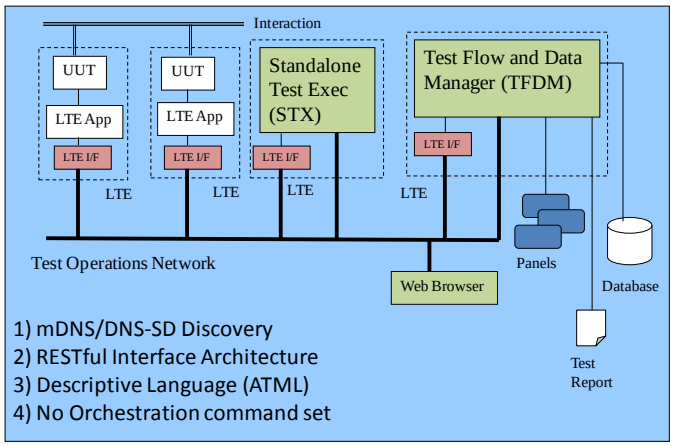


Figure 1. AHA Reference Topology

A Logical Test Element exposes its orchestration states as resources in a Restful web services interface. It also advertises its existence and capabilities for standard (mDNS/SD) service discovery. Standard business model resource groups can be defined to support specific test capabilities such as data gathering, event triggering and so on. An LTE could be anything from a web-cam, to a software simulation, to custom support software for an avionics subsystem, to COTS test equipment like a signal generator or oscilloscope. One useful LTE is a host computer itself, with the interface providing identification, performance, and processor loading statistics while also enabling applications to be started by a remote manager.

Two special case LTE concepts were prototyped to evaluate test flow with the AHA: the Standalone Test Executive (STX) and the Test Flow and Data Manager (TFDM). In addition to the standard LTE interface described above, these elements include DNS discovery software and a web client and are capable of discovering, monitoring, and commanding the other LTEs. Each has a specific role to play in the AHA test flow.

The Test Flow and Data Manager (TFDM) responsibilities include discovering and selecting LTEs to form a Test Configuration, configuring each LTE to a desired initial state, coordinating with the STX to execute Test Runs, and gathering and storing coordinated data from the LTEs. The TFDM also provides a central location for a Test Conductor to interact with multiple LTEs. The TFDM is data driven from the LTE resource metadata. In the implementation examples discussed in this paper, when an LTE is selected as part of an activity, a script creates a database table for it using the ATML metadata provided in the interface. The TFDM invokes test scripts, collects the data, and provides near real-time access to results. Although we anticipate a few sizes and shapes of TFDM, this code block is intended to be essentially write-once, developed by a skill set that is web- and database-oriented.

The Standalone Test Executive (STX) is intended to be composed by a subject-matter expert and contains specific knowledge of some of the LTEs, of the technique for running a specific test, and of the expected relationships among instrumented parameters. The STX represents captured expert knowledge. Obviously, a test procedure or outline might call out a sequence of various STX invocations. The STX itself generally provides for configuration and status through an LTE interface. For example, the STX might be given a time budget to ration, or it might calculate and report modeled ideal performance compared with measurements, or transfer functions or ratios. The STX is initiated and supported by the TFDM which provides environmental variables, and data logging, plotting, and reporting services.

These concepts were developed through several small-scale demonstration activities.

III. Concept Validations

In order to develop and demonstrate solutions for the most challenging aspects of the architecture, while demonstrating its flexibility, several small “proof” tasks were undertaken. Large-scale demonstrations were not possible or desirable in this design cycle, and software products were not finished out. The demonstrations were understood to be exploratory: disposable, unburdened by intellectual property concerns, and outside the critical path of other projects. They were conducted in an effort to identify best practices, and accumulate lessons learned. The intent was to expose the technology to a representative variety of applications and an assortment of operating environments and applications. All of these activities were conducted within the Avionic Systems Division of the Engineering Directorate at NASA’s Johnson Space Center.

A. Orchestration of Software Simulations

We demonstrated the use of the AHA interface to sequence, start up, discover, monitor, and shut down Trick simulations and EDGE (Engineering Dynamic Onboard Ubiquitous Graphics (DOUG) Graphics Environment) graphics applications. This activity used AHA (Figure 2) to orchestrate a distributed Orion abort-to-orbit test scenario split between JSCs Avionics Integration Environment (AIE) facility and the Reconfigurable Cockpit Simulation Facility which supplied hand controller (HC) hardware and cockpit displays.

An XSLT file was co-hosted with the ATML file in order to improve human readability when using a browser. We also began using Asynchronous Javascript and XML (AJAX) to improve display performance in browser interfaces. The LTE interfaces were executed on Linux systems and were distributed between the facilities. An AJAX orchestration

interface panel generated by the TFDM was accessed through a browser collocated with the operator cockpit.

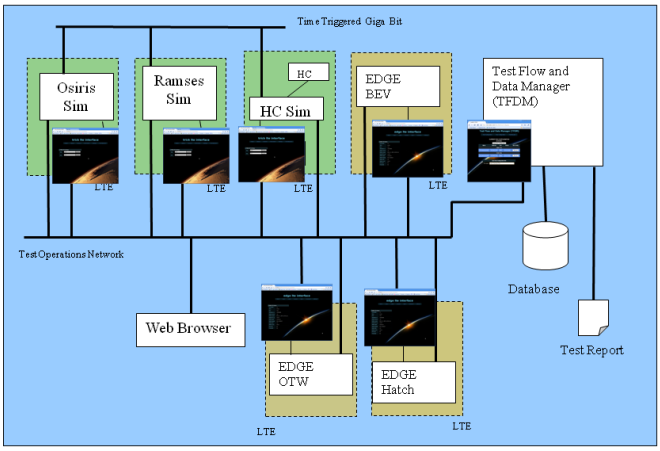


Figure 2. Orchestrating Simulations using AHA

B. Orchestration of a Parametric Sweep

The Electronic Systems Test Laboratory (ESTL) at JSC set up an off-line "Orchestration Sandbox" consisting of a simple communication link instrumented for Bit Error Rate, with clock jitter as a stimulus variable. This project reused pre-existing fully-developed LabVIEW applications running under Windows XP and the LTE interface connected to them through an ActiveX backend interface without altering existing LabVIEW code. In a parametric sweep, a stimulus is changed and allowed to settle, and then measurement statistics are settled during an "observation interval" before the data for the interval is recorded. Thus, the data is not plotted as a "strip-chart" against a time axis drawn from the same table, but instead data tables must be joined before the data is selected from multiple parallel tables. This simple task requires no more sophistication than a relational database offers provided that a common index exists.

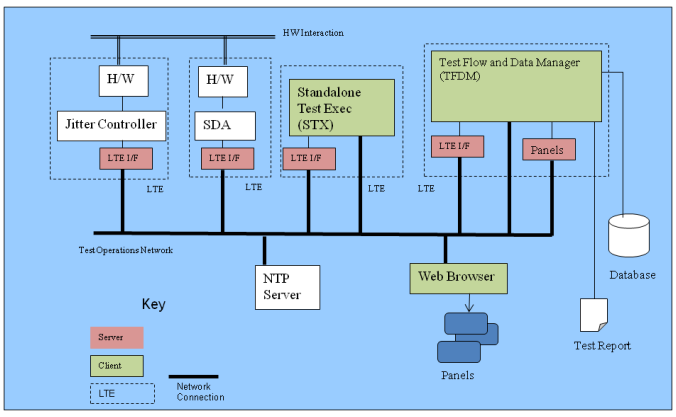


Figure 3. Parametric Sweep Orchestration using AHA

This activity (Figure 3) allowed us to refine the concept of the "STX," and it was here that we recognized that the "TFDM" needs to provide its own AHA interface. By exposing resources, the TFDM allows the STX to discover which of the LTEs visible on the network are selected as part of the activity, and the STX can prescribe when and what documentation the TFDM should collect. We can further see that this solution offers a natural approach to distributed testing, where each facility in a different location can have its own orchestrator, and an additional orchestrator can orchestrate the orchestrators. The same stacking technique might be used to scale a TFDM by dividing the workload instead of redeveloping database and network infrastructure to increase performance.

To ease the integration with LabVIEW, we experimented with using an Orchestration Virtual Instrument (OVI) hidden panel which could control a front panel as a user would. This concept allowed us to leave the finished LabVIEW panels and AHA LTE interface code alone. This concept was later generalized, but now a LabVIEW Template approach is making this extra layer vestigial. The OVI cannot be entirely eliminated because changing values through the LabVIEW ActiveX interface does not trigger "value-change" events as the keyboard does.

In working with LabVIEW we also stumbled over pop-up dialog boxes, and latched Booleans. At present, we simply avoid these. Error messages can be handled through a status-bar, logging time-tagged errors to a file, beeping, or other mechanism.

We were able to join the tables and plot the curve as it was being run, as well as overlay baseline prior data. For this activity we simply joined the tables based on time stamps truncated to the nearest second. Although this approach did support the demonstration, we would like to develop a more sophisticated and reliable technique using an additional table to associate records by observation interval.

Traditional approaches to test automation use extensive custom command sets. We were very pleased with the simplicity of resource-driven scripts, and the robust recovery of the test flow when manual interventions were required because the automation had wandered beyond limitations.

C. Mixed Avionics Hardware and Simulations

In an effort to shift to a more portable "road-show" format, we built a Portable Avionics Testbed Demonstrator using a laptop, a tablet, a Beagle board, an I/O pump, and a pair of hand controllers (Figure 4). This was a human-in-the-loop test, where an evaluator used a hand controller to perform a spacecraft docking.

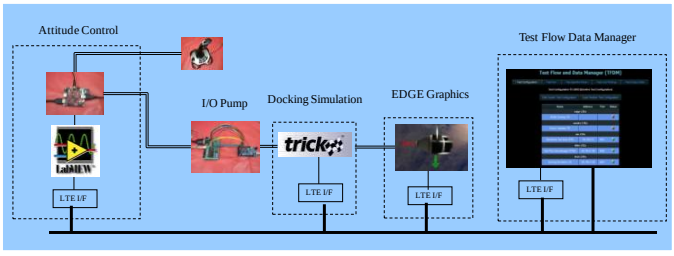


Figure 4. Orchestrating Mixed Hardware and Simulations

The Beagle board, standing in for a spacecraft controller, was configured and statused by a "Ground Support Equipment" LabVIEW application running under Windows on the tablet. The simulation and graphics packages were running under Linux. These modules could all be discovered and parameters from the controller and the simulation were stripped into the database.

D. Equipment Monitoring

For the Equipment Monitoring application we did not continuously log data and the topology does not include a TFDM (Figure 5). Essentially, a LabVIEW application monitored equipment in two racks (Fore and Aft) of hardware in the JSC Avionics Integration Laboratory (JAIL). An operator could monitor the LabVIEW control panel, but an STX also continuously monitored the panel in the background. As a capability demonstration, when a parameter would reach an alarm trip-point, the STX would point a webcam at the offending rack, and then email the out-of-range parameter value and the photograph to a responsible engineer.

The web camera we selected hosted its own web interface, providing a great opportunity to compare implementations; we were able here to directly integrate our interface with an off-the-shelf product using only the LTE Interface and no additional software. One advantage of the AHA web server interface is that through the use of hypermedia links it can play easily with an existing web interface on the AHA application software without getting in the way. We also demonstrated that we could re-host our interface onto a VxWorks embedded platform running LabVIEW, and we necessarily used a Hypervisor interface in place of the ActiveX connection we use with LabVIEW under Windows. (Our first-ever LabVIEW prototype used a DLL connection, but we don't recommend this more deeply integrated connection for LabVIEW). And so we now had connected our LTE Interface to socket, ActiveX, REST, and Hypervisor interfaces for data harvest.

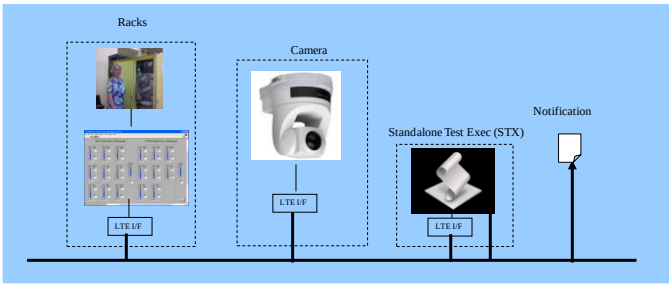


Figure 5. Equipment Monitoring using AHA

E. Supporting a Principle Investigator

We seized an intersection opportunity to support a Human Cognitive Technology Demonstration by removing our hand controller from the Portable Avionics Testbed Demonstrator and replacing it with a Brain Computer Interface (Figure 6). This allowed an evaluator to perform a hands-free docking task in support of an investigator.

At this stage, we added a Hyperic system monitoring application as an LTE that monitors health of our hosts. We also used an STX to provide the evaluator with some assistance, supervision, and feedback. An LTE interface was connected in front of the Microsoft Windows-based Emotive headset software using the Emotiv Software Development Kit. A TFDM AJAX interface was used for startup, shutdown, configuring data logging, and producing the ATML test results.

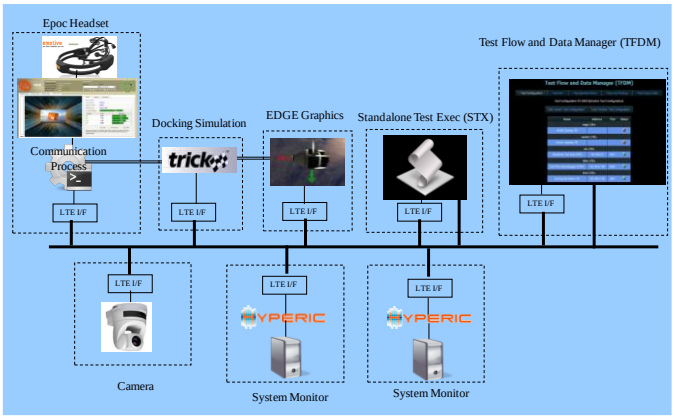


Figure 6. Configuring an Experiment using AHA

F. Code Cleanup and Code Generalization

Finally, we had an opportunity to work back through our code and try to incorporate a few of the lessons learned. In this process, we tried to generalize our LTE Interface software and improve the robustness of our prototype implementation of the architecture. We also made a first pass at constructing a LabVIEW Template package where the interface is always transparently present from the start of development. Additionally we prototyped some verification tools to exercise our interfaces repetitively while measuring performance and validating responses.

The LabVIEW Template development also intended to demonstrate that the LTE interface could also be used to host other useful features such as links to the GUI and auto-generated help files (harvested from documentation entered into the user interface), and a blog feature.

We finally modified our TFDM orchestrator to implement parallel threads so that an activity can flow around a non-responsive LTE. The orchestrator also supports multiple clients. Importantly, we implemented a caching architecture so that relatively bulky but static metadata need be retrieved from an LTE interface only once. To be effective, this means that the LTEs need to implement the "Expires" and "Cache-Control" headers already provided by HTTP in our protocol set. Our tests indicate this will offer significant performance improvement by reducing network traffic and sheltering LTE hosts.

Placing a blog feature in the LTE Interface package provides operators with a consistent and convenient method of journaling an activity so that notes can easily be collected together and compared. One application of course is that an operator can capture notes (timing and rationale for configuration changes, anomalies, observations, and conclusions) that are available later during analysis and reporting. But the blog is also a strategy for achieving and tracking software quality by standardizing and promoting communication between users and developers. The blog is implemented as an extra pair of resources in our interface, and so the feature need not be confined to user-oriented LTEs.

The blog feature of course is not implemented in ATML. It uses the Atom syndication format instead to create feeds of content entries that can be subscribed to using widely

available feed readers. The help files are composed in HTML. The LabVIEW GUI uses a browser plug-in downloaded automatically from National Instruments. Thus, we see no reason that we cannot co-host other XML formats with ATML in our interface. For example, our REST architecture "pulls" data, but we believe we could support XTCE stream definitions and links to XTCE-described streams. Further, we currently use only the TestResults and Common ATML schemas but the interface could host additional ATML documents.

Throughout our architecture validation tasks, we expected that we would standardize our resource tree. We have concluded this is both an unnecessary and undesirable constraint, and instead recommend a hypermedia layout. The hypermedia layout will improve our backward compatibility as we make changes ("future-proofing"), will improve performance by separating data from metadata, and will simplify scripting as parameters are duplicatively grouped into functional "collections" instead of singularly categorized into a tree.

We are splitting our protocol set because we believe much of it has versatility extending to many other usages. Our formulation of mDNS/SD, Rest, HTTP, and hypermedia we are relabeling as an "mREST" foundation. Our formulation of specific orchestration features combined with ATML becomes the "testing" application of "mREST." We believe this will simplify our interface definition and expand the opportunity for collaboration.

G. Scale-to-Zero Bench Test

Often a hardware or firmware developer will write a simple application in a high-level language like LabVIEW to control and status their unit during development. A part of our concept of operations for the LabVIEW Template has been that we could transparently embed our machine-facing interface in a user-facing application from the beginning of development, and the designer would find it useful enough that the interface itself would receive some functional verification long before the software appeared at an integration activity.

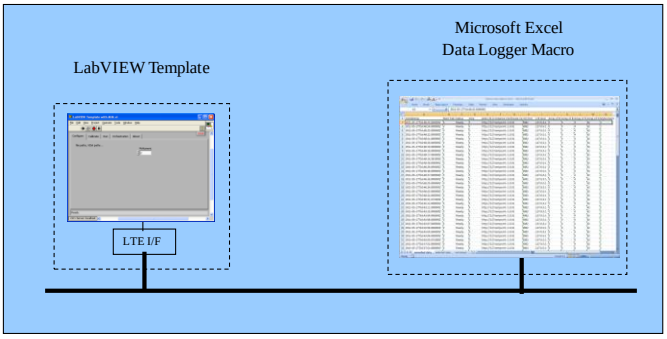


Figure 7. Zero-Infrastructure Data Logging using AHA

As a demonstration then, Microsoft Excel was co-hosted with the LabVIEW Template (Figure 7). The URL for the LTE Interface was pasted into Excel as the location of an external XML data source file. Formulas were used for convenience, to identify elements to be captured. And finally a macro was composed from a recording. The 16-line macro refreshed the data once a second for ten seconds, each time inserting a row in the spreadsheet and pasting the linked data.

Even array elements were captured this way. Of course the spreadsheet could also be used to analyze the data, calculate figures of merit or compare to models, and maintain plots. And a formula result could be used to control the program flow so that change-only data is logged.

But the point of this exercise was to demonstrate that no extra hardware, middle-ware, documentation, or even special skill is required to begin exploiting the power of the API. Implementation on a trivial scale accomplishes worthwhile performance-logging work.

IV. Overall Lessons

Shallow internal connections were a goal because they minimize the cost of adding and maintaining the interface and maximize the possibility of retrofitting the interface. We see it will be possible to accumulate a set of tools for quickly installing the interface or building it in from the beginning of an LTE development.

RESTful architecture concepts were found to greatly simplify implementation and reduce coupling between test elements. Thinking of test integration and test flow in terms of resource manipulation instead of large command sets was a paradigm shift. We think it holds promise for simplifying testing design, scripting and implementation. Another paradigm shift we encountered was using discovery techniques and hypermedia instead of rigid interface control documents to reduce the cost and effort of maintaining interface compatibility between test elements. We believe this has promise in reducing overall lifecycle costs for testing in the NASA environment and has application to other areas requiring asset management at NASA and in industry.

V. Areas for Future Development

Although ATML is a rich and adolescent (approaching maturity) schema set, we remain concerned that our concept of operations, where an LTE may be Test Equipment in one situation and a Unit Under Test during a calibration, may require accommodations. To date we have not found institutional support for engaging specific NASA experts who could mitigate these concerns by evaluating ATML against other completed study conclusions. Areas of potential concern include a comparison with NExIOM (NASA Exploration Information Ontology Model) [2] to identify gaps, a comparison with MDX (multidimensional expressions used for data-mining of OLAP databases by business intelligence), a comparison with XTCE (xml Telemetric and Command Exchange) to determine interoperability (we strongly suspect translation losses here cannot be avoided), and special requirements for live operating environments and distributed testing conducted by teams with many affiliations.

We would like to finish construction of our LabVIEW template to promote some meaningful deployment, allow us to use larger topologies in our next development spiral, and clean up our portability between Windows and Linux.

We still need to prototype a sophisticated transient response test with event-driven flow and data aggregation (trials, points, curves, surfaces). This will push our tools significantly forward and help uncover advanced issues with data formats and labeling.

Soon we will need to prototype a procedure executor (as-run or re-run). One area of interest will be to see how resource-oriented test flows, such as those implemented by our STX, can be described as ATML test requirements. We will add trivial orchestration features that promote deployment, and investigate more advanced features that promote scalability.

We will also begin involving more data product consumers to evaluate our formats and processes, begin fielding our tools to assist research projects, and then begin injecting our technology set into facilities and projects.

In the process we will continue to refine and stabilize our standard collections and standard resource definitions. For example, most LTEs will want to provide a "health" collection where some resources like "not_safe" are standardized and others simply adhere to conventions so that an operator can use software tools to manage a larger number of software applications.

VI. Conclusion

The Automation Hooks Architecture initiative reduces the cost of technology and science production by mobilizing equipment, people, and knowledge through the use of common tools plugged into open-standards interfaces.

We believe a spiral approach to affordable and effective data integration is prudent: set up all of the pieces and look at how they fit together before returning to invest more heavily in developing quality and features in each of them.

Our effort is by nature collaborative as we seek to identify a simple but broadly powerful formulation of interfaces and tools for data collection and reduction. Advancement and distributed use of this approach is encouraged as the next step strategies for larger scale adoption as a standard.

We currently rate this interface as Technology Readiness Level 5.

Acknowledgment

This endeavor required focusing many kinds of nonintersecting experts on a multi-faceted problem.

We would like to thank Robert Klinger and Craig Ross, supporting the Maestro orchestrator development at Marshall Spaceflight Center, for their concept of stacking copies of a single orchestrator development to perform distributed testing. Maestro is the orchestrator developed for the Constellation Program and is built around a Program-specific interface.

Joan Zucha, ESCG, provided the Soft Decision Analyzer prototype to make bit error rate measurements in the Orchestration Sandbox, a trivial use of the SDA.

Kwaku Nornoo, ESCG, introduced the concept of an Orchestration Virtual Instrument to assist with LabVIEW integration.

Juan Uribe, ESCG, and Ellen Keulemans, ESCG, provided the equipment-monitoring applications for JAEL.

We thank Jason Ekstrand for producing the Portable Avionics Testbed Demo during a summer internship at JSC.

Our principle investigator for Human Cognitive Technology is David Fletcher, also known as "Jedi Master."

We would also like to thank Chris Gorringe, ATML (IEEE SCC20 TII) Chair, for supporting us with an open dialog on ATML usage.

And none of our proofs would have been possible without the faith placed in us and guidance provided by David Lee and Andre Sylvester.

References

[1] C. A. Lansdowne, J. R. MacLean, et. al., *Automation Hooks Architecture Trade Study for Flexible Test Orchestration*, ISBN 978-1-4244-7960-3, Autotestcon Proceedings, Sep. 2010.

[2] P.J. Keller, *Preparing for Semantic Technology in SOA*, SOA Symposium Government and Industry Best Practices, April 2010.

Automation Hooks Architecture for Flexible Test Orchestration

Concept Development and Validation

AutoTestCon 2011
September, 2011

Chatwin Lansdowne
John MacLean
Christopher Winton
Patrick McCartney

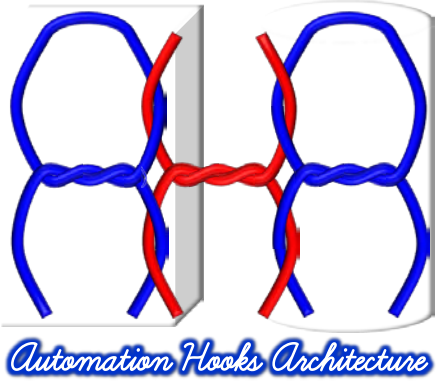
Custom Software is Doing More Work Than Ever

- New tools put software production in the hands of subject-matter experts
- Control panels on COTS products are being replaced by software
- Dedicated stand-alone test stations are being built around proprietary solutions

These developments are **user-centric data-islands**

- Can we define a software and data architecture that will **integrate on a macro-scale**…
- That we can **produce and use on a micro-scale**…

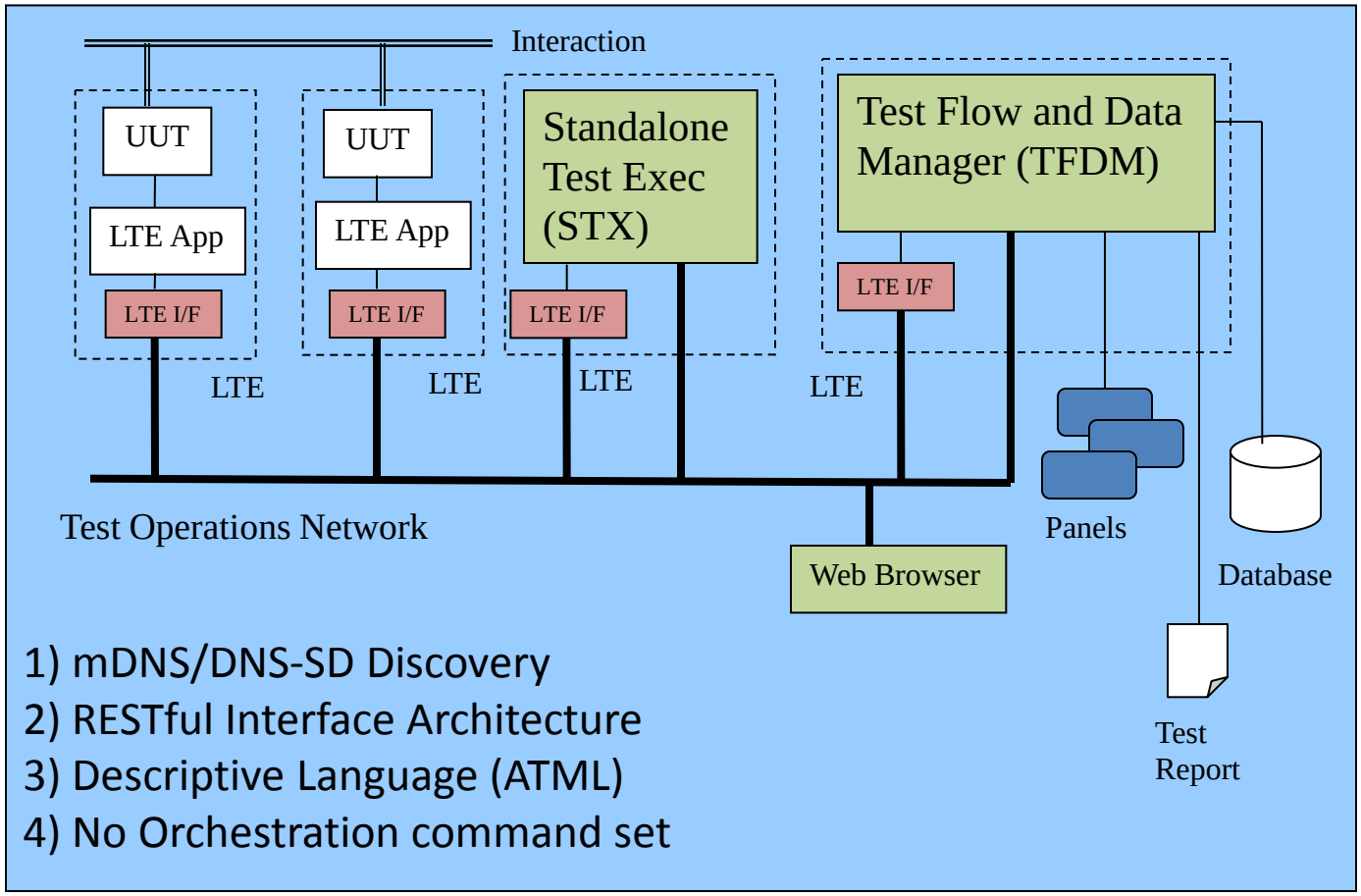
…it's just Standards



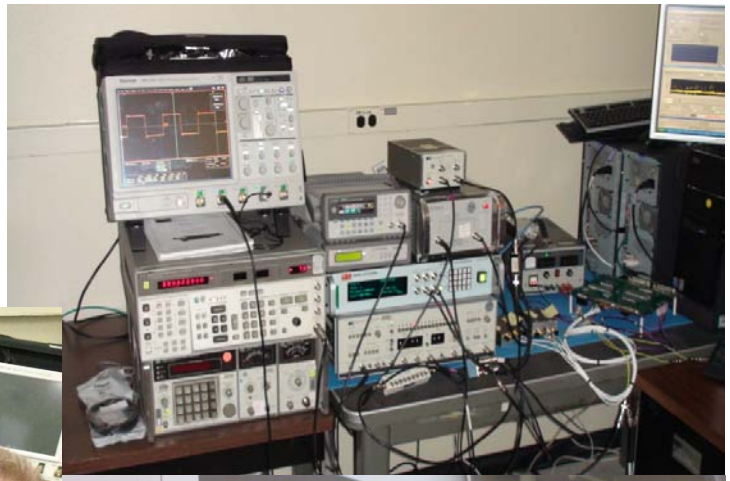
Study Result presented AutoTestCon 2010

- **REST Architecture**
  - elsewhere used: Microsoft Robotics, webcam, Web of Things
  - "pull" data flow
  - powerful control with two simple commands
  - can host support files and links– interface definitions, requirements, theory of operation, links to streaming data and web-based GUI
- **Advertised**
  - elsewhere used: LXI, consumer products
  - enables unmanaged dynamic IP address and port assignments
- **HTTP**
  - elsewhere used: web browsers, web pages, Excel
  - standardizes messaging, error messages, cache controls, message compression, security
  - TCP/IP-based (adjustable time-out)
- **xml**
  - elsewhere used: migration to xml, although not painless, is the path being taken by architecturally-aware organizations like Microsoft and DoD
  - standardizes communication of metadata, which we're using to create tables in modern xml-enabled databases
- **xml:ATML (IEEE 1671)**
  - elsewhere used: coming feature in DoD procurement specs
  - standardizes units, arrays, time zone– and opens exciting opportunities for COTS tools and radically different engineering work flows
- **Orchestration features**
  - Available scheduled data collection and configuration changes
  - Health and Status

AHA Reference Topology

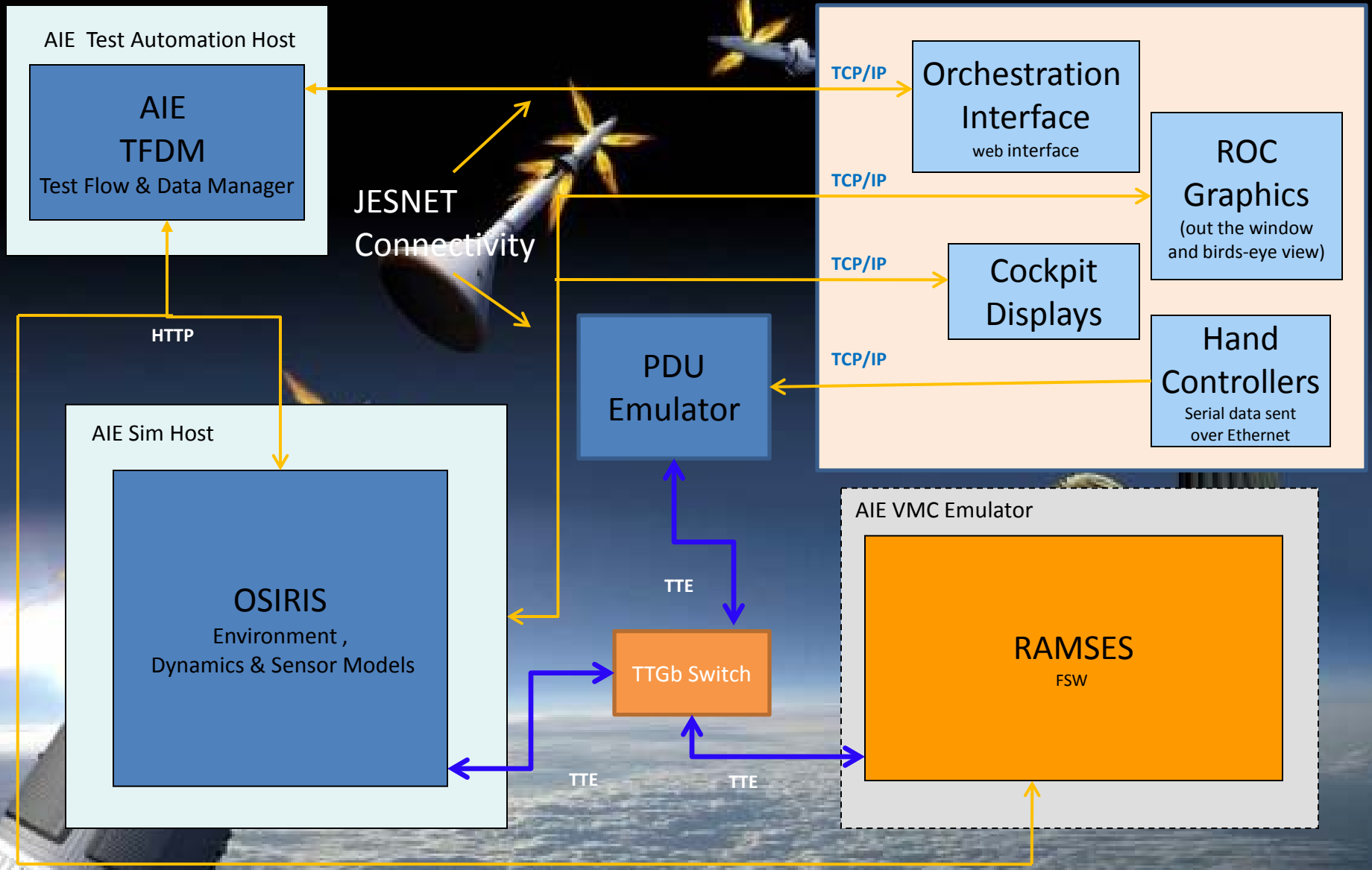


Concept Validations

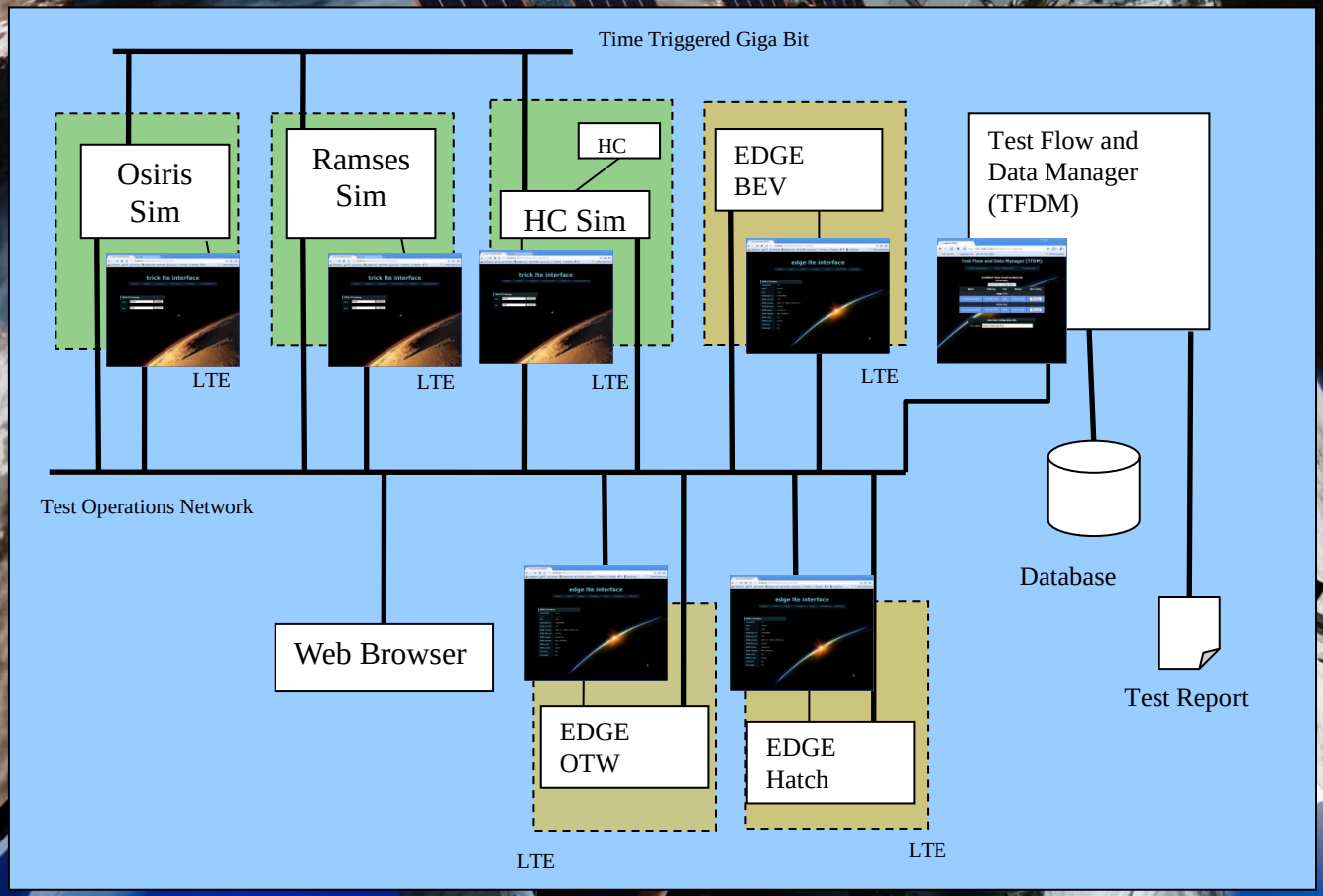


- Orchestrating Software Simulations
- Orchestrating a Parametric Sweep
- Avionics Testbed
- Equipment Monitoring
- Supporting a Principle Investigator
- Code Cleanup and Generalization

Integrated AIE/ROC Orion Demo

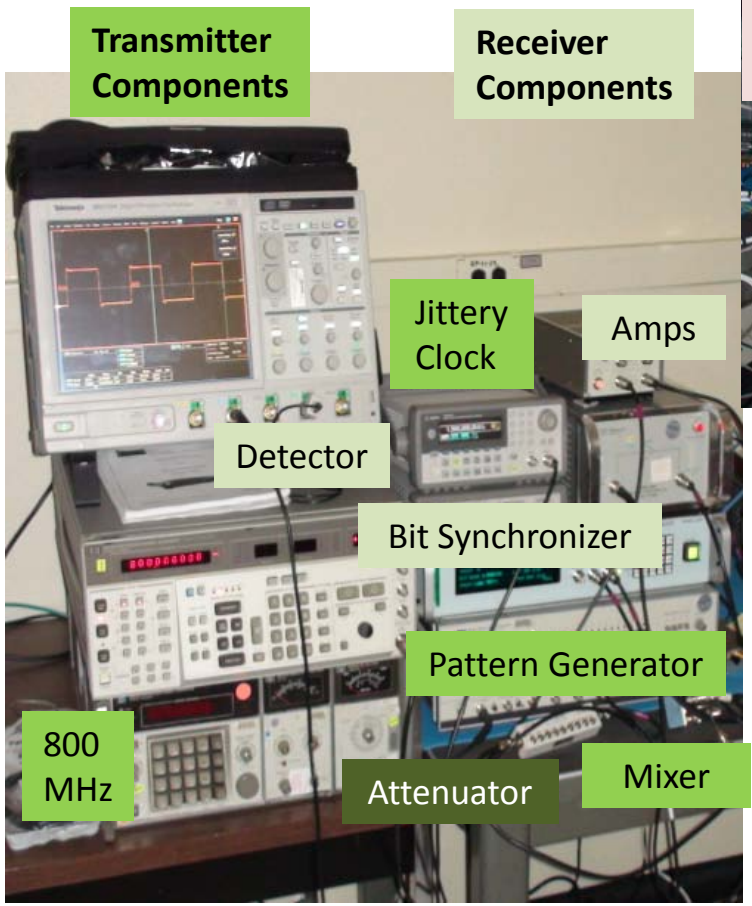


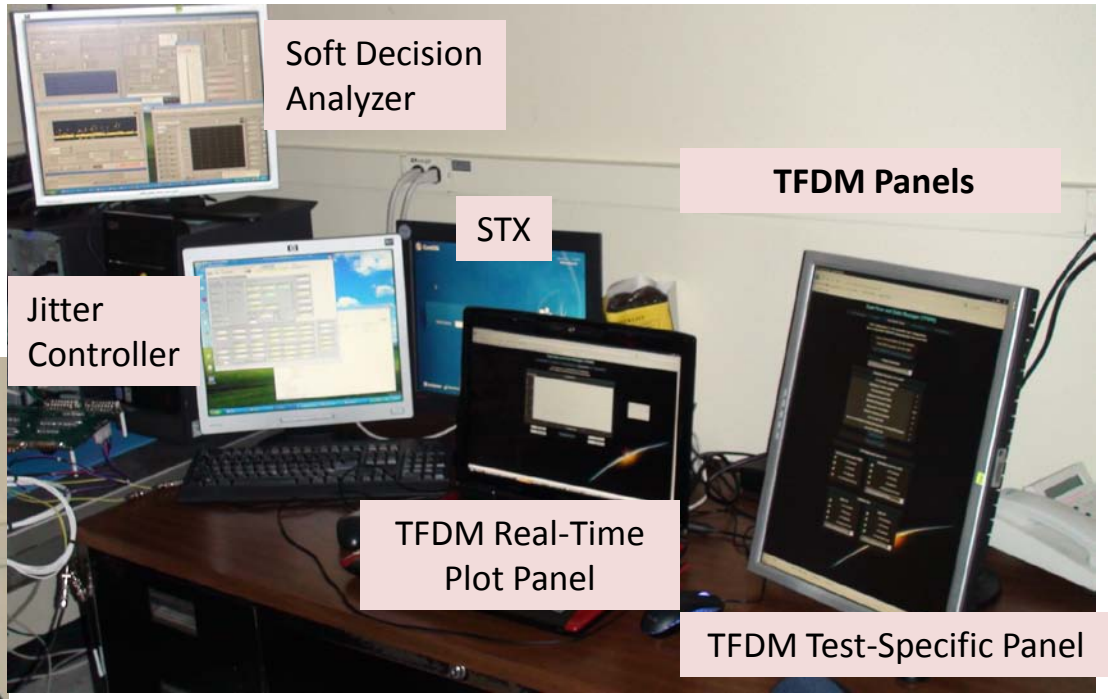
Integrated AIE/ROC Orion Demo



- Startup, shutdown, and monitoring of AIE and ROC LTEs from ROC
- Improved LTE browser interface with XSLT
- Added EDGE interface
- Upgraded TFDM prototype to Asynchronous JavaScript and XML (AJAX)

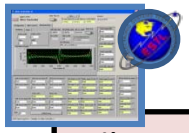
Parametric Sweep Configuration





- Prototyped automation of parametric test flow (sweep a curve)
- Developed Standalone Test Exec concepts
- Added interface to existing ESTL LabVIEW test software with no modifications
- Added additional database and reporting capability

Automated BER v



Jitter Controller Platform

SCPI AHA

AHA

Ethernet Switch

Jitter Controller SCPI

E5810A LAN/GPIB Gateway

GPIB

HP33250 Arb Generator

HP8663A Signal Generator

HP8081A Pattern Generator

800MHz

OOK

Kay

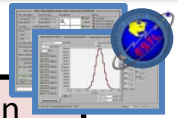
Ref Data

Ref Clock

TRANSMITTER

CHANNEL

AHA



Soft Decision Analyzer Platform

Test Data

Test Clock

XD3A Diode Det

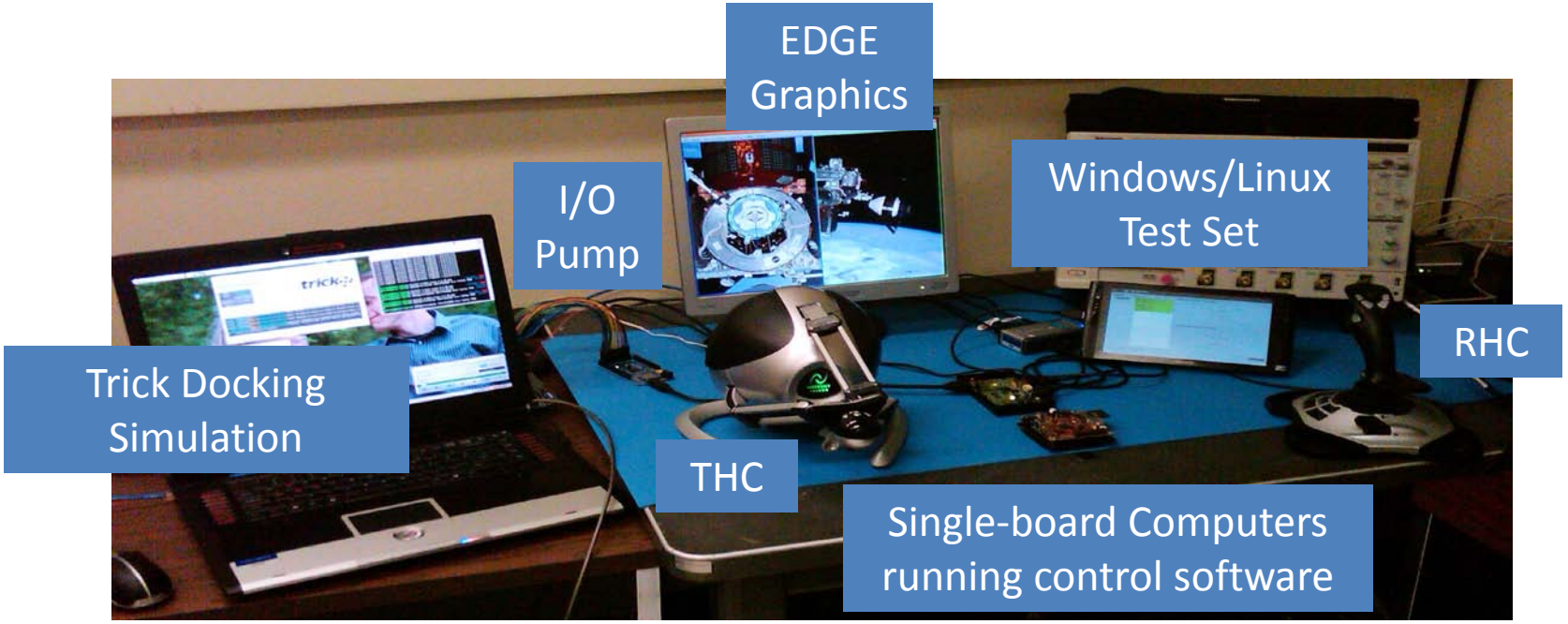
C-Core

Data

GDP 2265D Bit Synchronizer

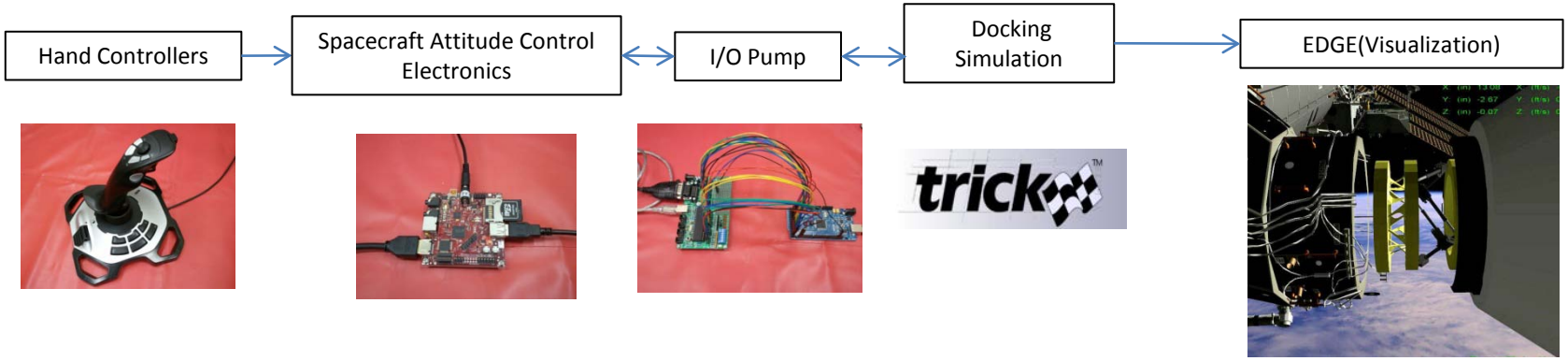
RECEIVER

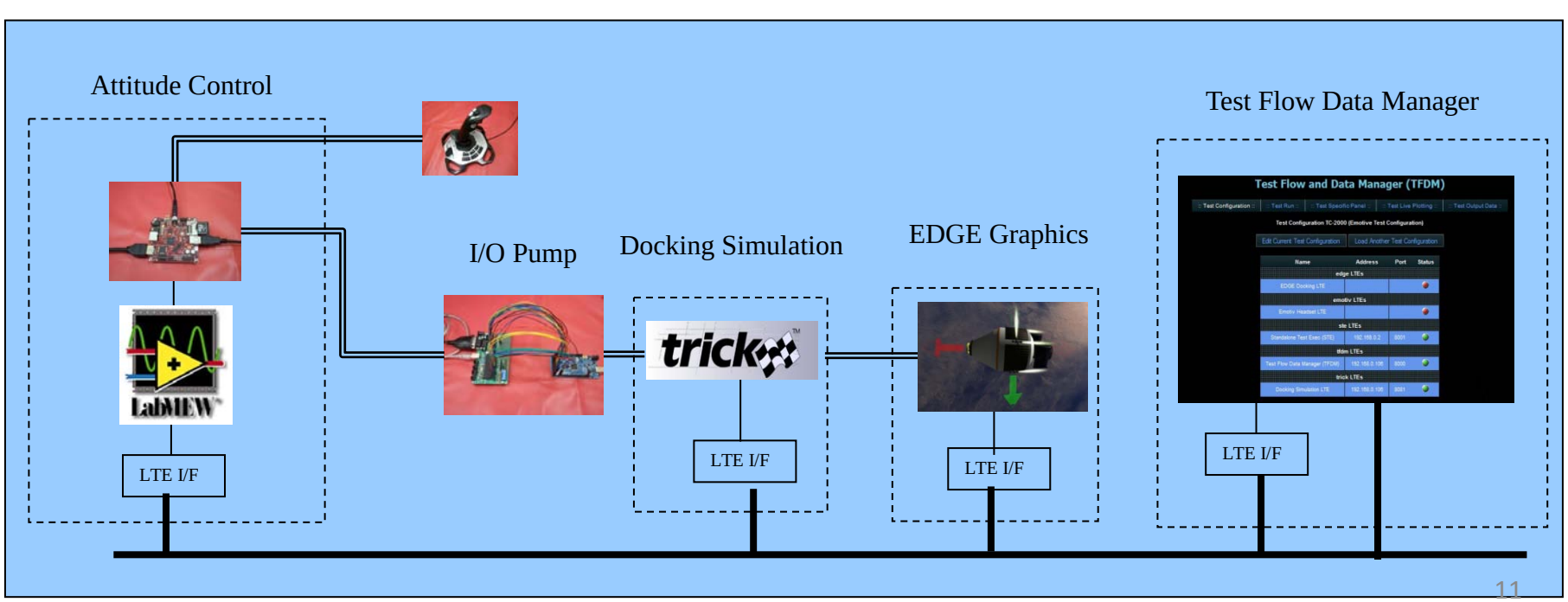
Portable Development Test bed:
Mixed Avionics Hardware, Simulations, and Crew



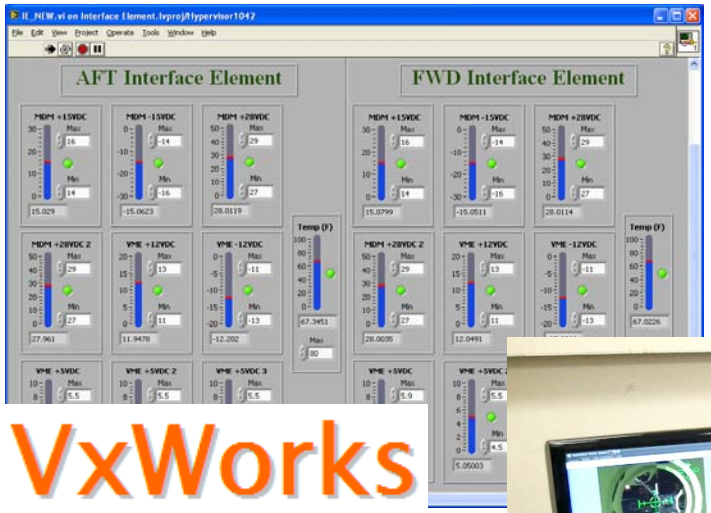
- Low-cost orchestration development test bed
- Leveraging off of a functional LIDS docking model developed for CxTF
- Single board computers and I/O pump to mimic avionics hardware
- Uses results from three intern projects

Portable Development Test bed:
Mixed Avionics Hardware, Simulations, and Crew





Iteration 4: Sep-Oct/10

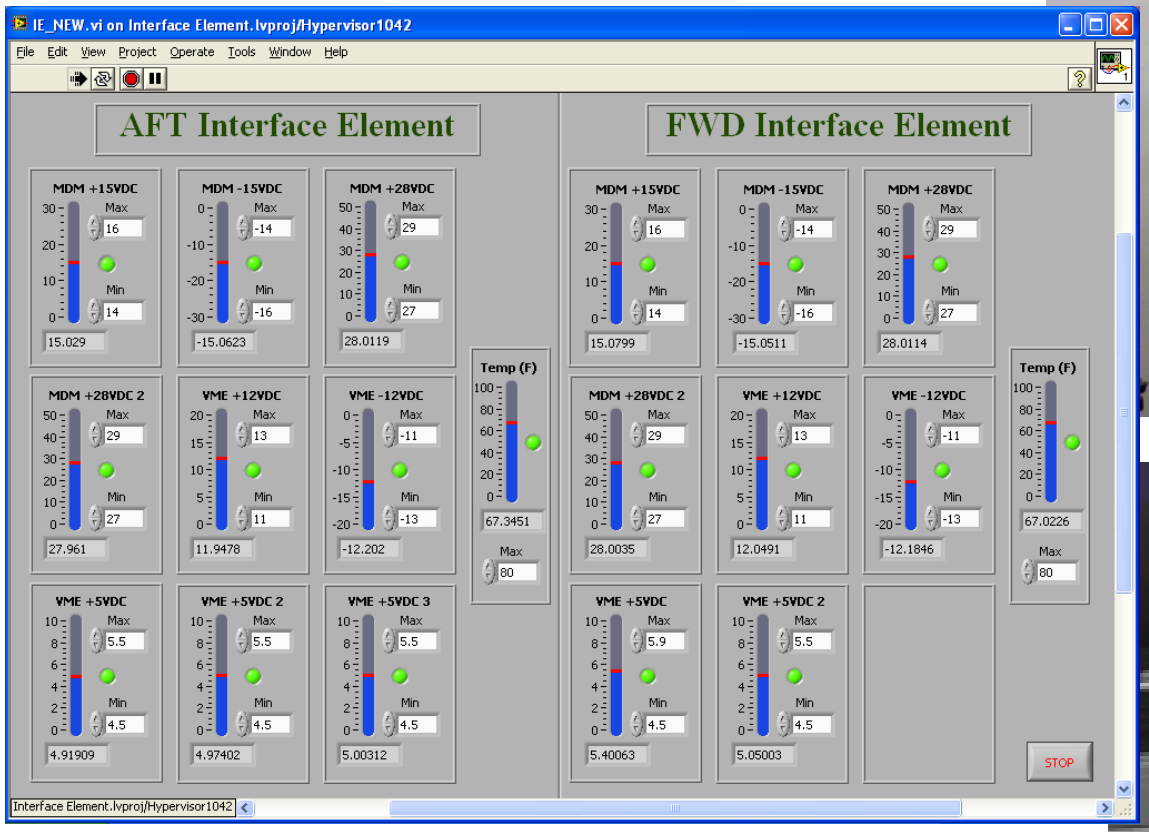


VxWorks

- Additional LTE Interfaces
- Human-in-the-loop Orchestration

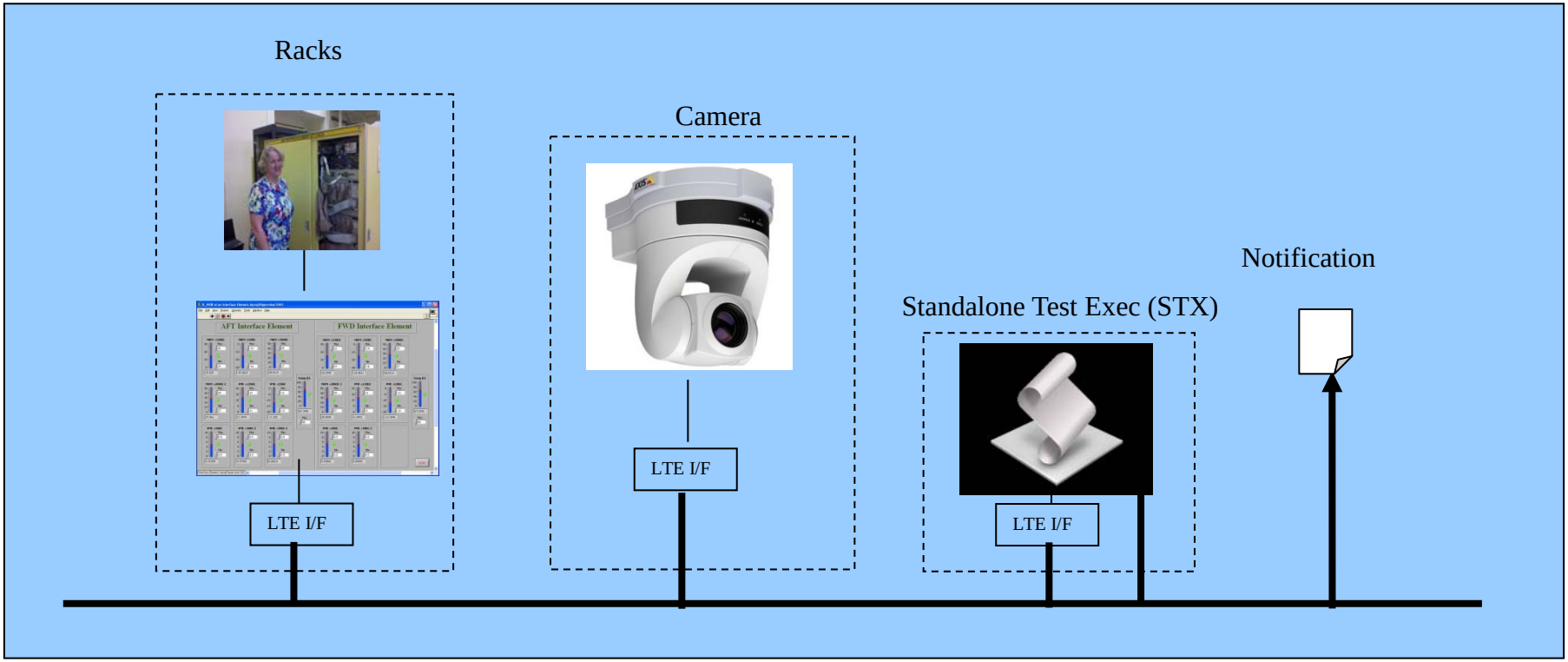
JAEL Rack Monitoring Application

Iteration 4: Sep-Oct/10



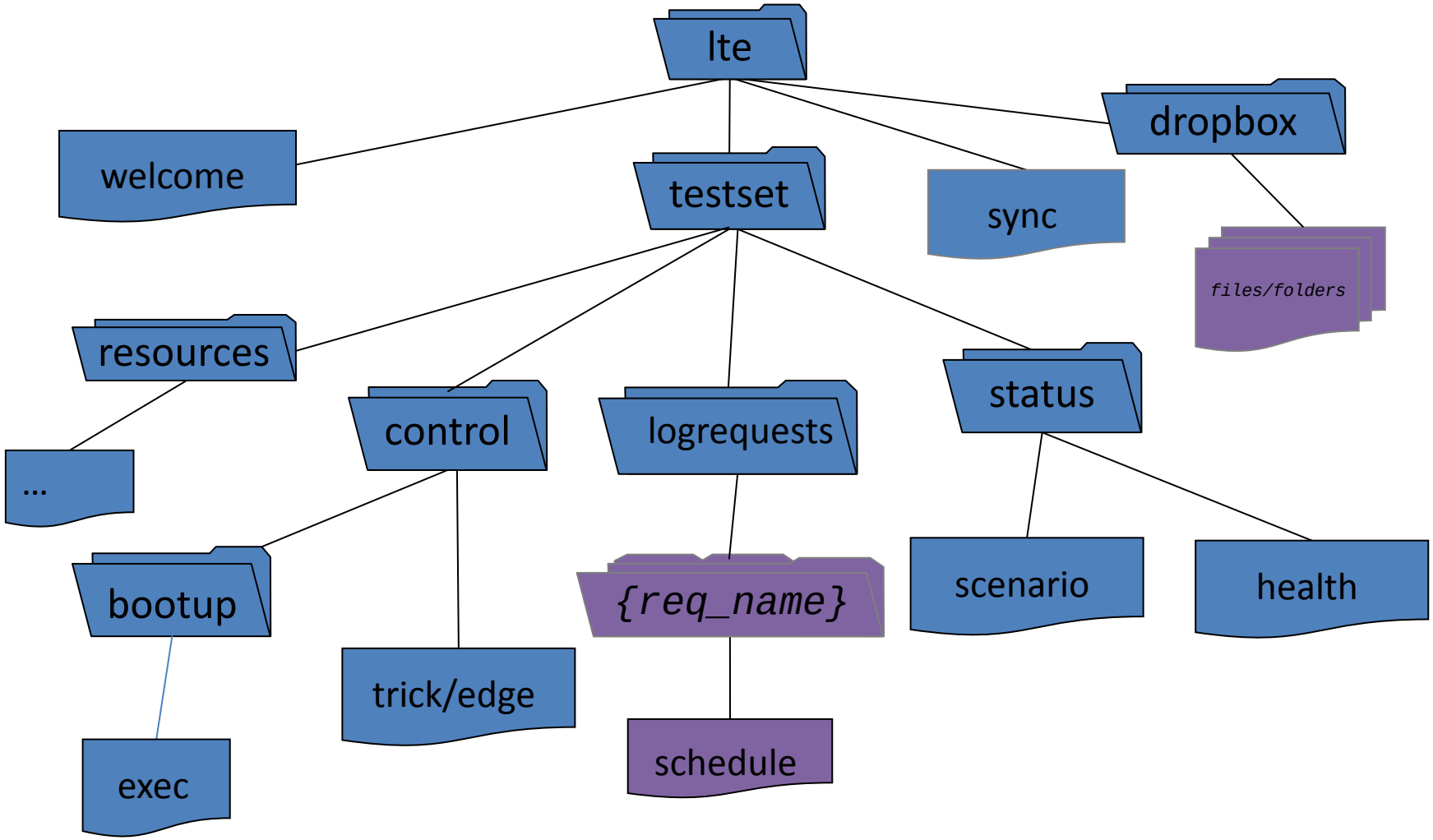
JAEL Monitoring Application

Iteration 4: Sep-Oct/10



- Prototyped interface to Hypervisor/VxWorks version of Labview
- Use of LTE/IF with off the shelf hardware (netcam)
- Use of STX to monitor LTE I/F (No TFDM)

LTE Resource Example (AIE)



Human-in-the-Loop Emotive Evaluation

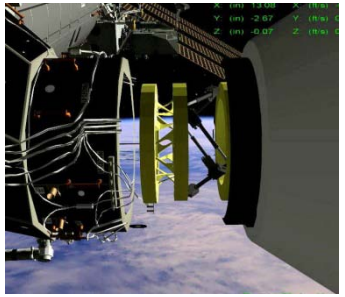
Iteration 4: Sep-Oct/10

Human-in-the-Loop Emotive Evaluation

Iteration 4: Sep-Oct/10

Emotiv Epoc → Emotive EmoEngine → Emotive to Trick Communication Process → Docking Simulation → EDGE

trick™

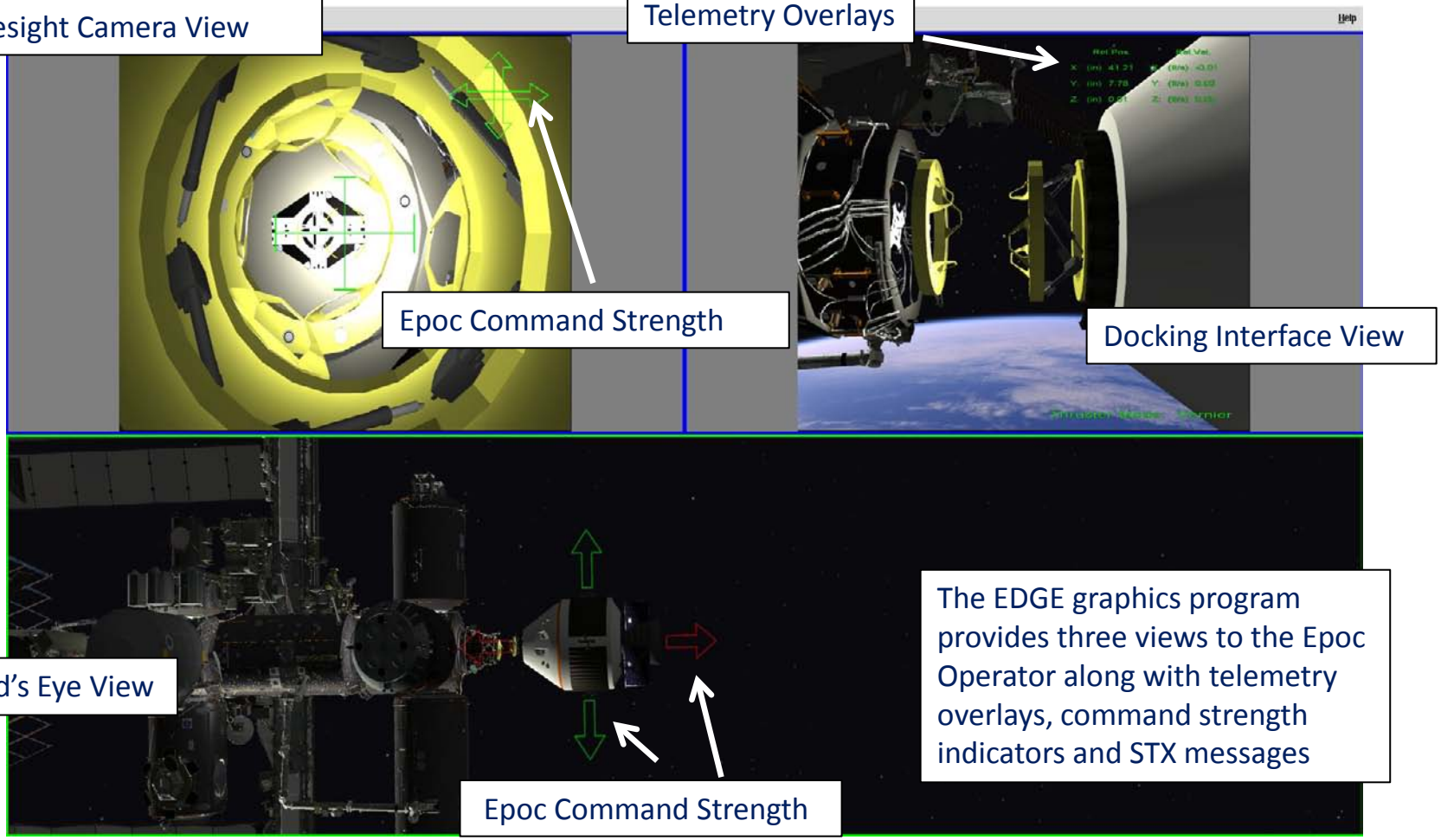


Epoc Headset provides thruster control for
- Forward/Back
- Starboard/Port

Because Epoc headset has only four cognitive outputs, an auto pilot provides thruster control for
- Rotational Degrees of freedom
- Up/Down
- Attitude Hold (upon request)

Human-in-the-Loop Emotive Evaluation

Iteration 4: Sep-Oct/10



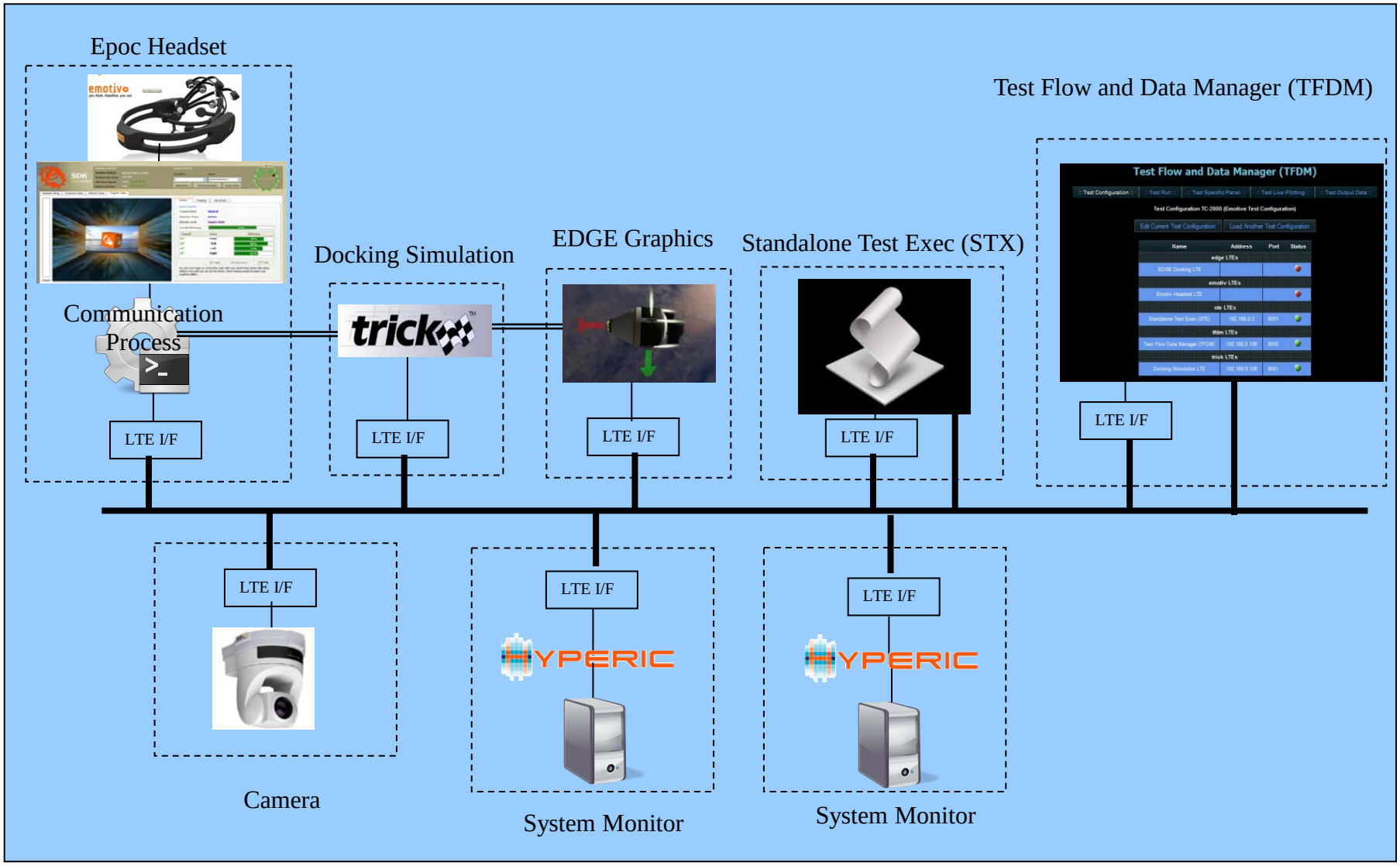
Human-in-the-Loop Emotive Evaluation

Iteration 4: Sep-Oct/10

- Prototyped Orchestration
  - Bring up and down software and simulations
  - Configure headset and evaluation parameters for each run
  - Initialize simulation state and pass/fail monitoring
  - Monitor docking performance and collect statistics
  - Assist participant with paper-pilot activities
    - keep spacecraft in evaluation envelope
    - switch coarse-vernier
    - extend LIDS ring/ turn on docking light
  - Determine and report pass/fail of run
  - Organize test results
  - Monitor health of computers and software

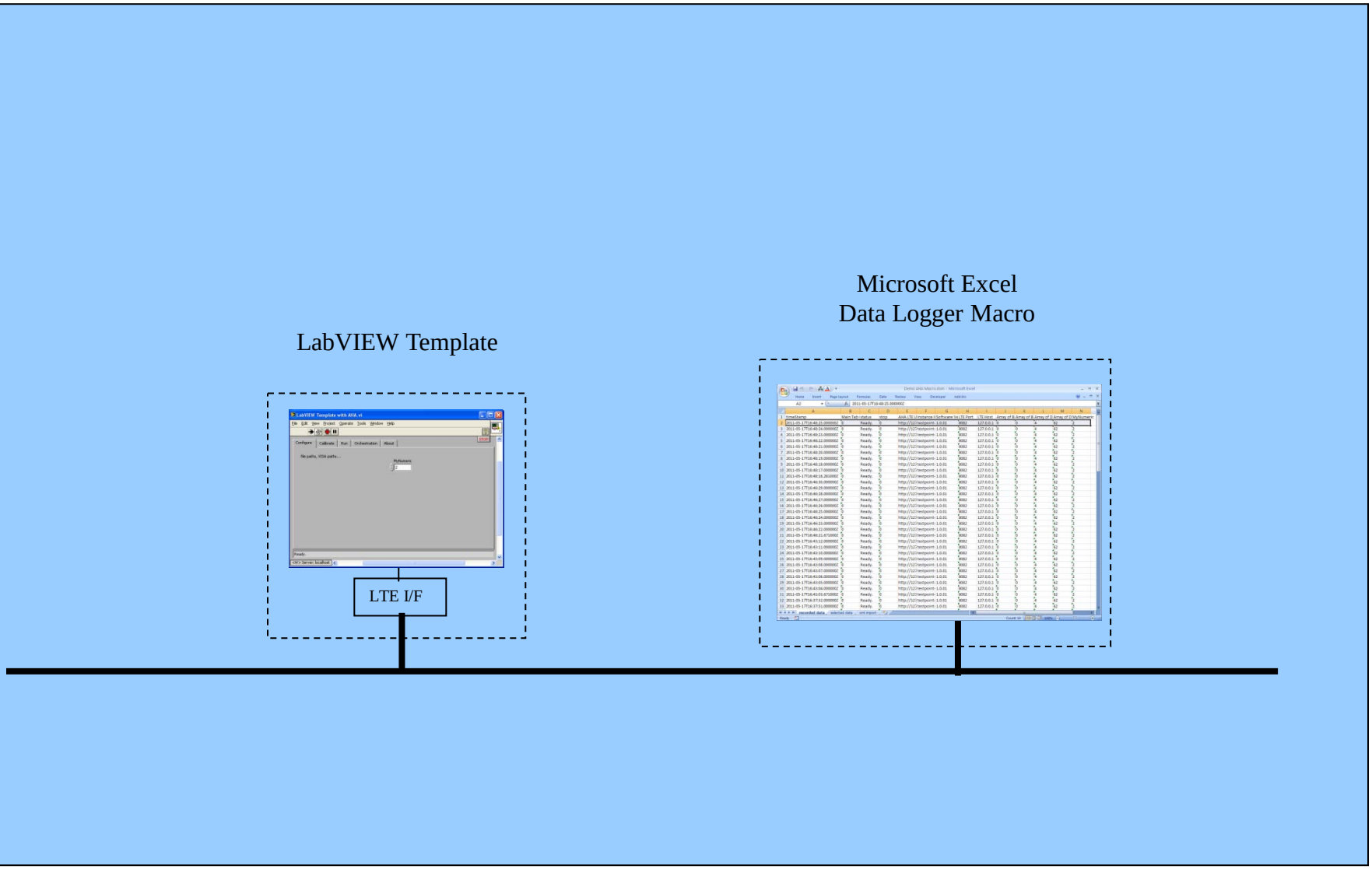
Human-in-the-Loop Emotive Evaluation

Iteration 4: Sep-Oct/10

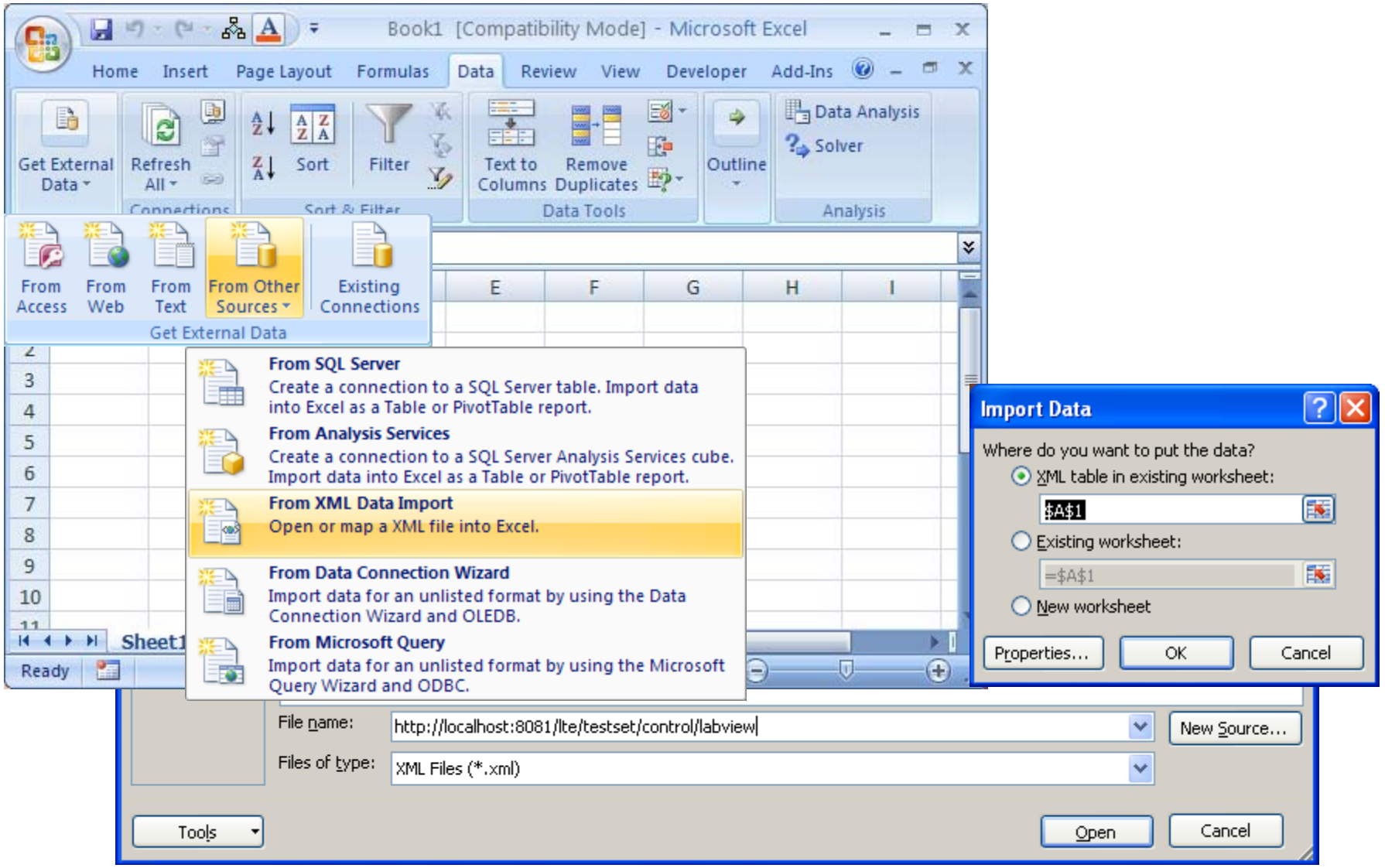


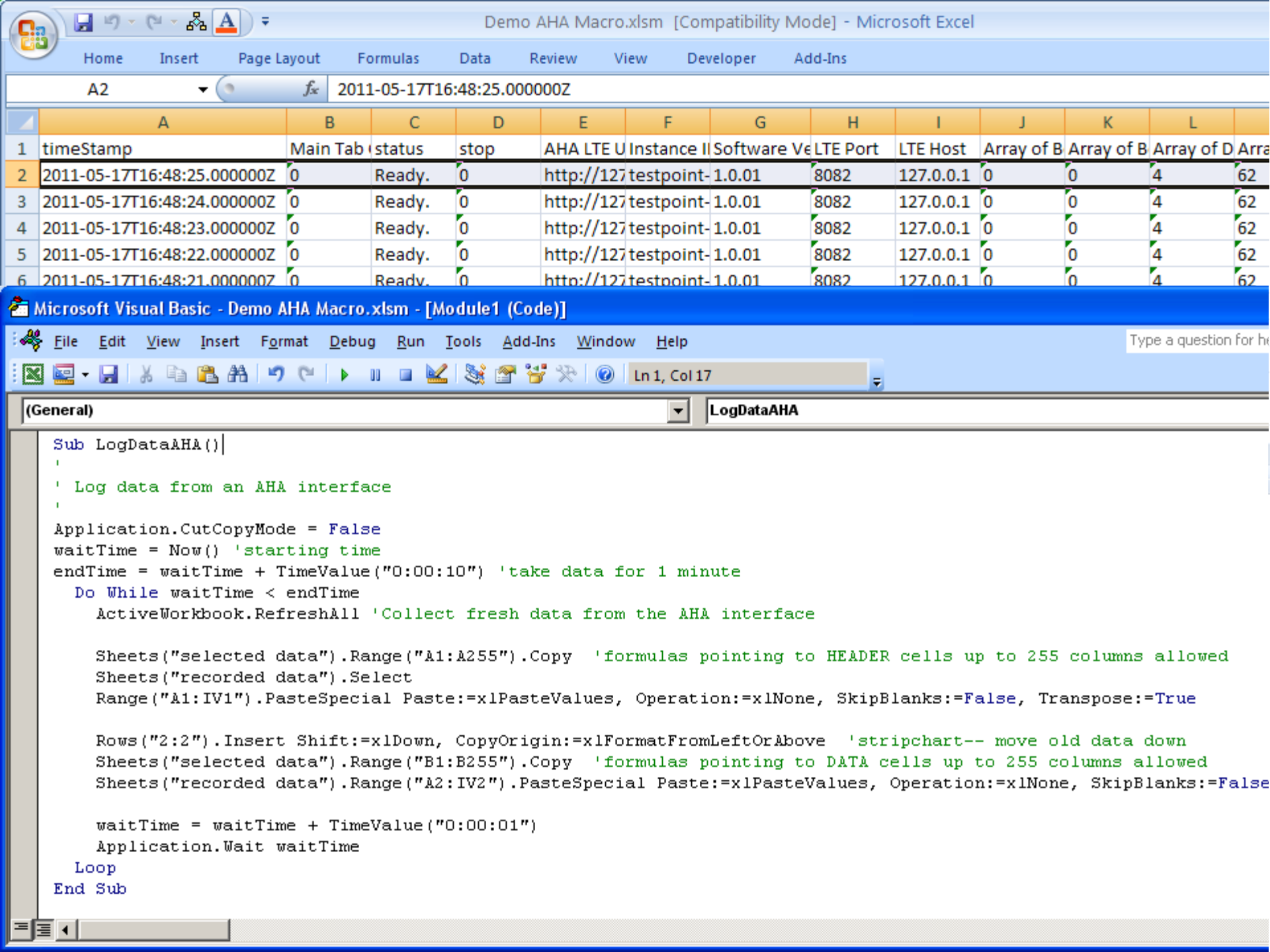
Bench Test Evaluation

Post-Iteration 4: May/11



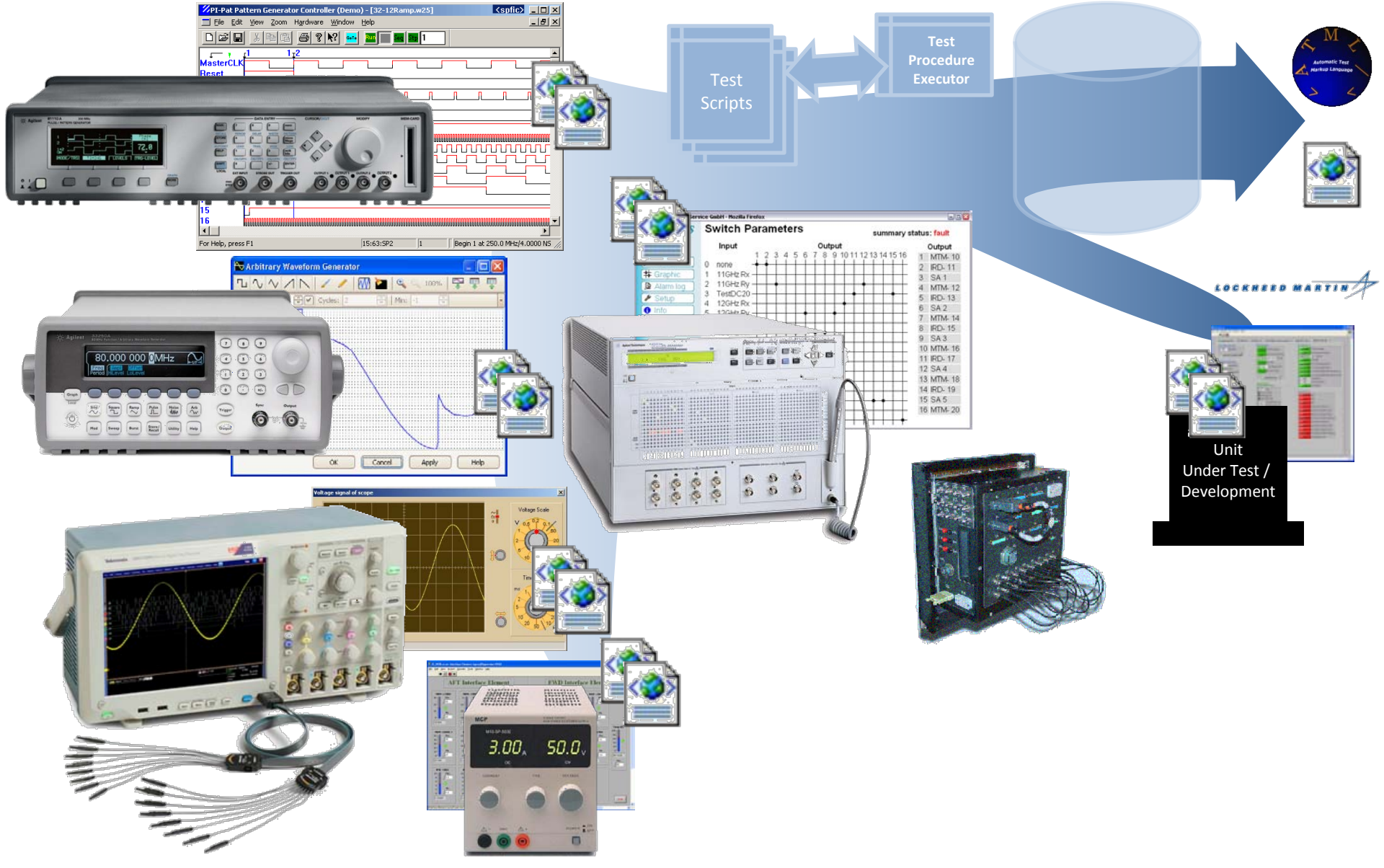
Linking to the API from Excel




Demo AHA Macro.xlsm [Compatibility Mode] - Microsoft Excel
Home Insert Page Layout Formulas Data Review View Developer Add-Ins
A2
2011-05-17T16:48:25.000000Z
timeStamp Main Tab status stop AHA LTE U Instance II Software Ve LTE Port LTE Host Array of B Array of B Array of D
2011-05-17T16:48:25.000000Z 0 Ready. 0 http://127 testpoint- 1.0.01 8082 127.0.0.1 0 0 4 62
2011-05-17T16:48:24.000000Z 0 Ready. 0 http://127 testpoint- 1.0.01 8082 127.0.0.1 0 0 4 62
2011-05-17T16:48:23.000000Z 0 Ready. 0 http://127 testpoint- 1.0.01 8082 127.0.0.1 0 0 4 62
2011-05-17T16:48:22.000000Z 0 Ready. 0 http://127 testpoint- 1.0.01 8082 127.0.0.1 0 0 4 62
Microsoft Visual Basic - Demo AHA Macro.xlsm - [Module1 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 1, Col 17
(General)
LogDataAHA
Sub LogDataAHA()
'
' Log data from an AHA interface
'
Application.CutCopyMode = False
waitTime = Now() 'starting time
endTime = waitTime + TimeValue("0:00:10") 'take data for 1 minute
Do While waitTime < endTime
ActiveWorkbook.RefreshAll 'Collect fresh data from the AHA interface
Sheets("selected data").Range("A1:A255").Copy 'formulas pointing to HEADER cells up to 255 columns allowed
Sheets("recorded data").Select
Range("A1:IV1").PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks:=False, Transpose:=True
Rows("2:2").Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove 'stripchart-- move old data down
Sheets("selected data").Range("B1:B255").Copy 'formulas pointing to DATA cells up to 255 columns allowed
Sheets("recorded data").Range("A2:IV2").PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks:=False
waitTime = waitTime + TimeValue("0:00:01")
Application.Wait waitTime
Loop
End Sub

ATML: Think Outside the Rack

ATML: Think Outside the Rack



More Kinds of Tests

- Engineering Properties of Materials: Outgassing, tensile strength, thermal, etc.
- Subsystem EMI/EMC
- Subsystem Vibration and Thermal
- System (internal) Integration mixed with Simulations
- End-to-End Systems integration: Proof of Concept to FEIT, MEIT
- Training Sessions
- and the list goes on…

AHA is a “Straw-man”

- We’re no longer just flagging the problem, we’re trying to piece together a candidate *plug-and-play* solution
- We examined a lot of choices and prototyped the ideas we thought were most promising

Does anyone have a better idea?

BACKUP

Criteria for a Software Architecture

- **Platform-independent:** *everyone can use their own appropriate operating system, language, and tools*
- **Inexpensive:** *quick to add, easy to learn, simple to test and maintain*
- **Rapid Assembly:** *quick and easy to integrate and troubleshoot*
- **Data Integrity:** *minimal translations, meta-data capture, archive-quality product, restore by write-back, simplified analysis and reporting*
- **Self-Contained:** *the instructions and documentation are in the interface*
- **Open Standards:** *architectural interfaces can be specified by referencing published non-NASA standards*
- **Non-proprietary:** *support multiple COTS vendors for robustness*
- **Open Source:** *supporting user communities are active and tools and chunks are widely available, widely tested, and widely reviewed*
- **Web-based:** *works with the tools you carry in your pocket*
- **Data-Driven:** *the code can be stable, only support-files change*
- **Low-infrastructure:** *stand-alone capable, minimal reliance supporting infrastructure and staff IT experts*
- **Modularity:** *operations can proceed with broken modules*
- **Durability:** *maintenance is not required for legacy bought-off modules on legacy platforms*
- **Retrofit to compiled code:** *sometimes we have to work with what's available…*
- **Convergence:** *a direction observed in aerospace, test, DoD, and consumer products industries and communities*
- **Versatility:** *the more useful it is, the wider it will be implemented*
- **Scalability:** *scale up– or down to one*

Restoring the Viability of NASA's Facilities and Developments

The need for Modern Standards and Practices

- Common tools and Portability of skills
- Agility: Flexibility **and** Speed
  - Fewer standing, dedicated capabilities
  - Reuse/redeployment of assets and people
- Increased quality and detail in Data Products
  - No typos
  - More statistical significance and resolution
  - Ability to locate and interpret "cold" data
  - Analyzing "sets" not "points"

Why Think Outside the Command Set?

- The **state** of the configuration is always available to read, write, record, or restore
- The HTTP command and error-message sets already have extremely broad acceptance
- Move from Command-Driven to Data-Driven– with REST, the interface is self-describing. Scripting and orchestrating are accomplished by manipulating collections of discoverable "resources."

Why Think Outside the Rack?

- Distributed tests: box level, sub-system integration, system integration, systems integration… fielded systems, dispersed instrumentation, distributed simulations
- Do the ATML document modules support these concepts? There is a “configuration under test” and “element description” rather than a “test set” and a “UUT”.
- Is ATML sufficiently modular that vendors of generic test equipment could post an “element description” which can be picked up and folded into a “test set description”? Post ATML results at an interface just as they provide LXI? Do ATML documents “roll up”?

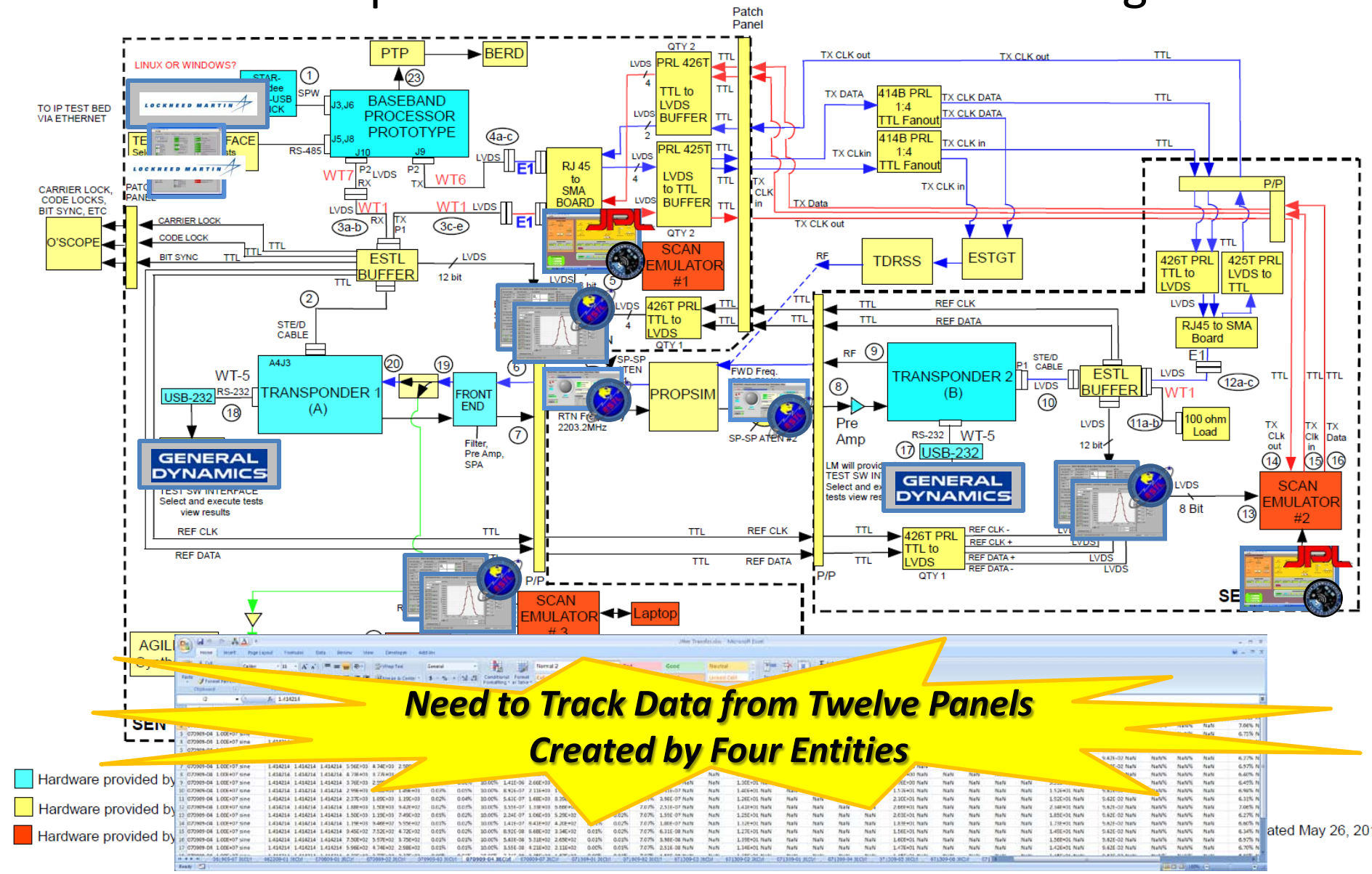
Why Do I Like ATML?

- Basic metadata for measurements: Units, Unit (statistic) Qualifier, Tolerance, Resolution…
- Standardized time, complex data structures like arrays
- Classified markings
- Contact information
- Potential for widespread use in aerospace and beyond aerospace
- Comes with an architecture for test automation, and COTS tools are already showing up (this is an intriguing trend we're still contemplating).

… ATML is a rich and thoughtful schema, and it's a standard produced by a significant world organization and backed by significant enterprises

NASA Use-Case Example

Orion Transponder – Baseband Processor Integration



Tentative Comments for ATML

Interoperability with XMLA and mdXML

- XML for Analysis... is an industry standard for data access in analytical systems, such as OLAP and Data Mining. XMLA is maintained by *XMLA Council* with Microsoft, Hyperion and SAS being the official XMLA Council founder members.[1]
- MultiDimensional eXpressions... XMLA specifies **MDXML** as the query language. In the XMLA 1.1 version, the only construct in MDXML is an MDX statement enclosed in the <Statement> tag.

Aggregation Clues in ATML

urn:IEEE-1671:2009.02:Common

- <xs:attributeGroup name="**UnitAttributes**">
- <xs:annotation>
<xs:documentation>**In nearly all ATS use cases, strictly limiting units of measure to SI or English units is restrictive. In numerous cases, it is desirable to qualify a unit with an additional text string, e.g., Peak-to-Peak or RMS for voltage measurements. This attribute group allows for the inclusion of a standard SI unit of measure (as defined in IEEE Std 260.1 [B18]), a nonstandard unit of measure, and a qualifier thereto. Name Type Description Use nonStandardUnit c:NonBlankString Any nonstandard unit not already defined in IEEE Std 260.1 Optional standardUnit c:StandardUnit When used, this attribute shall contain only a unit of measure defined in IEEE Std 260.1 Optional unitQualifier c:NonBlankString A textual qualifier that is to be applied to the attribute of either the standardUnit or nonStandardUnit. Examples: RMS or Peak-to-Peak for a standardUnit of volts. Optional NOTE—If one is not sure if a particular unit being utilized is standard or nonstandard, assume it is nonstandard, and represent it as a nonStandardUnit.**</xs:documentation>
</xs:annotation>
<xs:attribute name="**standardUnit**" type="**c:StandardUnit**" use="**optional**" />
<xs:attribute name="**nonStandardUnit**" type="**c:NonBlankString**" use="**optional**" />
<xs:attribute name="**unitQualifier**" type="**c:NonBlankString**" use="**optional**" />
</xs:attributeGroup>

Aggregation features in MDX

- Avg function (MDX)
- Count function (MDX)
- Sum function (MDX)
- Min function (MDX)
- Max function (MDX)
- Median function (MDX)

I don't see "RMS" or "Product"…

- VarP function (MDX)
- StdDevP function (MDX)

- In practice, it appears that "measures" in a "fact table" are used to control aggregation.
- **Can the "fact table" be generated from the ATML Test Results?**

XMLA Data Types

- Arrays– format seems not to be standardized, user gets to decide and then user interprets
- Sets– array (ordered collection) of tuples

```
([Time].[Fiscal].[Month].[August],
[Customer].[By Geography].[All Customers].[USA],
[Measures].[Sales])
```

- Tuples– essentially structs, collections, clusters…

```
{([Measures].[Sales], [Time].[Fiscal].[2006]),
([Measures].[Sales], [Time].[Fiscal].[2007])}
```

ATML **Collection** Complex Type

<xs:complexType name="Collection“>
<xs:annotation> <xs:documentation>The Collection complex type shall be the base type for XML schema elements intended to contain multiple data values, i.e., unordered sets of values, ordered vectors of values (with the order of items in the vector being represented by the order of c:Collection/Item child elements), or collections of named values, also known as records (with the names being represented by the name attribute of the c:Collection/Item child element).
</xs:documentation> </xs:annotation>
<xs:sequence> <xs:group ref="c:DatumQuality"/>
<xs:element name="Item" minOccurs="0" maxOccurs="unbounded">
<xs:annotation><xs:documentation>Base type: Extension of c:Value Properties: isRef 0, content complex The Collection/Item child element shall contain an individual data value or vector. This child element is recursive; thus a Collection/Item may be a collection of data values or vectors.
</xs:documentation></xs:annotation><xs:complexType>
<xs:complexContent>
<xs:extension base="c:Value“>
<xs:attribute name="name" type="c:NonBlankString" use="optional">
<xs:annotation><xs:documentation>A descriptive or common name for the individual data value or vector.</xs:documentation></xs:annotation>
</xs:attribute>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="defaultStandardUnit" type="c:StandardUnit" use="optional">
<xs:annotation><xs:documentation>This attribute shall contain a unit of measure as defined in IEEE Std 260.1™ [B18].</xs:documentation></xs:annotation></xs:attribute>
<xs:attribute name="defaultNonStandardUnit" type="c:NonBlankString" use="optional“>
<xs:annotation><xs:documentation>This attribute shall contain any nonstandard unit, not already defined in IEEE Std 260.1.</xs:documentation></xs:annotation></xs:attribute>
<xs:attribute name="defaultUnitQualifier" type="c:NonBlankString" use="optional“><xs:annotation><xs:documentation>A textual qualifier that is to be applied to the attribute of either the standardUnit or nonStandardUnit. Examples include RMS and Peak-to-Peak for a unit of volts.</xs:documentation></xs:annotation></xs:attribute>
</xs:complexType>

Using “Live” ATML

- We find that we would like to be able to tag ATML data with a range or list of expected values. In a “live” setting, this allows us to screen entries, or provide a user with a pick-list.

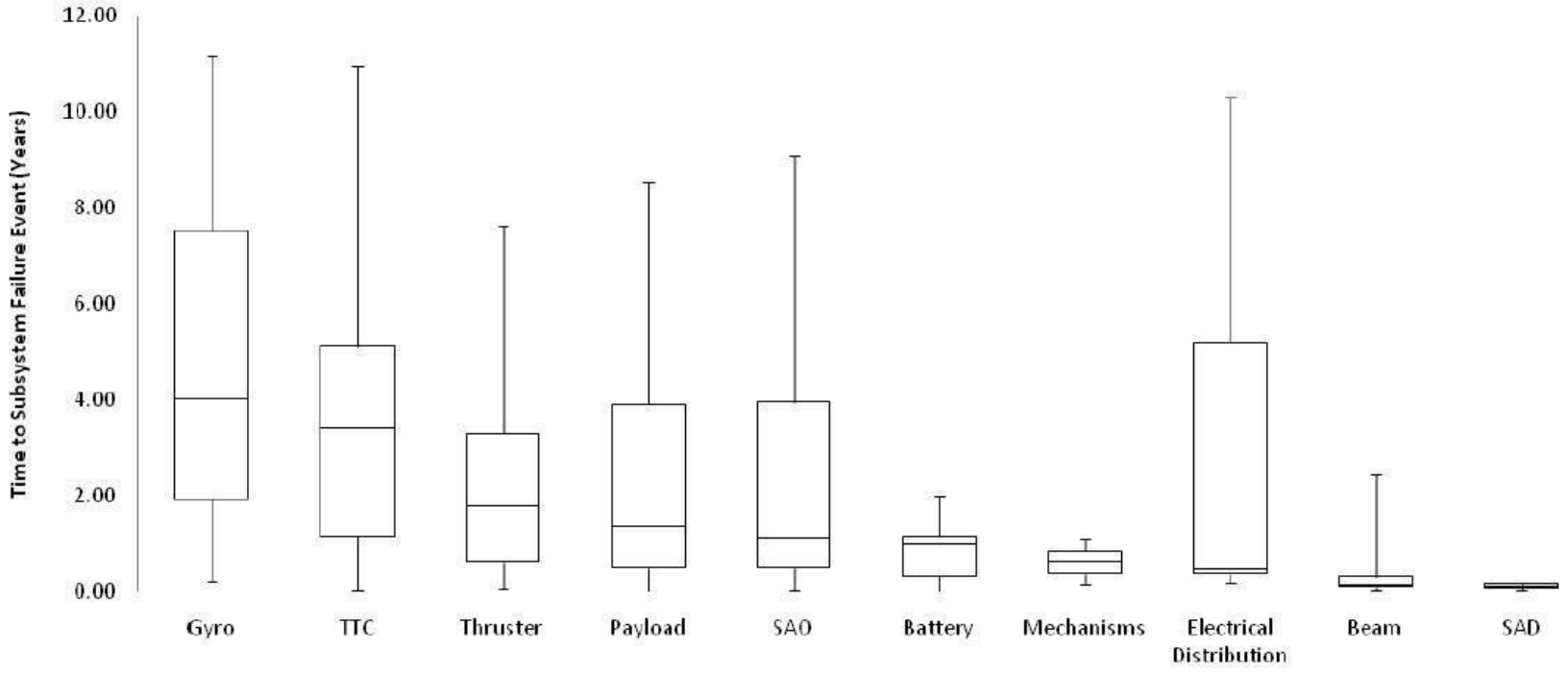
Tolerance Tags assume symmetry

- <xs:group name="**DatumQuality**">
…
- <xs:sequence>
…
- <xs:element name="**ErrorLimits**" type="**c:Limit**" minOccurs="**0**">
- <xs:annotation>
<xs:documentation>**Base type: c:Limit Properties: isRef 0, content complex The DatumQuality/ErrorLimits child element shall contain the error limits.**</xs:documentation>
</xs:annotation>
</xs:element>
- <xs:element name="**Range**" type="**c:Limit**" minOccurs="**0**">
- <xs:annotation>
<xs:documentation>**Base type: c:Limit Properties: isRef 0, content complex The DatumQuality/Range child element shall contain the range.**</xs:documentation>
</xs:annotation>
</xs:element>
- <xs:element name="**Confidence**" type="**xs:double**" minOccurs="**0**">
- <xs:annotation>
<xs:documentation>**Base type: xs:double Properties: isRef 0, content simple The DatumQuality/Confidence child element shall contain the required confidence.**</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:group>

Example of asymmetric uncertainty: Skewed Binomial Trials

- Example: Observe 1 million packets sent, one is lost. "Measured" loss rate is 1E-6.
  - If I run the experiment again, I might lose 2, or none.
  - 95% confidence interval extends +457% to the high side, but -457% to the low side is not even sane. Actual is about -76%.
- Example: Observe 1 million packets sent, zero lost. "Measured" loss rate is 0.
  - The experiment actually means I can say with 95% confidence that the loss rate is below 3E-6. It *could* be zero, but we don't know.

Example of Asymmetric uncertainty: Time to Fail



	Gyro	TTC	Thruster	Payload	SAO	Battery	Mechanisms	Electrical Distribution	Beam	SAD
Maximum	11.18	10.96	7.64	8.53	9.08	2.00	1.08	10.31	2.42	0.18 (64)
3rd Quartile	7.52	5.12	3.30	3.90	3.95	1.14	-	5.18	0.32 (117)	-
Median	4.02	3.42	1.78	1.36	1.10	0.99 (360)	-	0.48 (174)	0.13 (47)	-
1st Quartile	1.93	1.15	0.60 (220)	0.50 (182)	0.50 (181)	0.33 (121)	-	0.38 (138)	0.09 (34)	-
Minimum	0.19 (69)	0.03 (12)	0.06 (22)	0.01 (2)	0.02 (7)	0.00 (1)	0.15 (55)	0.18 (64)	0.03 (10)	0.03 (12)

Figure 4: Time to Subsystem Failure Event (Years).

Source: *Epidemiology of Satellite Anomalies and Failures: A Subsystem-Centric Approach*, Rachel A. Haga and Joseph H. Saleh, IEEE Aerospace Conference, March 2011

Standard units

- <xs:attributeGroup name="**UnitAttributes**">
- <xs:annotation>
<xs:documentation>**In nearly all ATS use cases, strictly limiting units of measure to SI or English units is restrictive. In numerous cases, it is desirable to qualify a unit with an additional text string, e.g., Peak-to-Peak or RMS for voltage measurements. This attribute group allows for the inclusion of a standard SI unit of measure (as defined in IEEE Std 260.1 [B18]), a nonstandard unit of measure, and a qualifier thereto. Name Type Description Use nonStandardUnit c:NonBlankString Any nonstandard unit not already defined in IEEE Std 260.1 Optional standardUnit c:StandardUnit When used, this attribute shall contain only a unit of measure defined in IEEE Std 260.1 Optional unitQualifier c:NonBlankString A textual qualifier that is to be applied to the attribute of either the standardUnit or nonStandardUnit. Examples: RMS or Peak-to-Peak for a standardUnit of volts. Optional NOTE—If one is not sure if a particular unit being utilized is standard or nonstandard, assume it is nonstandard, and represent it as a nonStandardUnit.**</xs:documentation>
</xs:annotation>
<xs:attribute name="**standardUnit**" type="**c:StandardUnit**" use="**optional**" />
<xs:attribute name="**nonStandardUnit**" type="**c:NonBlankString**" use="**optional**" />
<xs:attribute name="**unitQualifier**" type="**c:NonBlankString**" use="**optional**" />
</xs:attributeGroup>

Standard Units

6.3 Examples of symbols for limited character sets

Table 4 shows many examples of how the foregoing principles may be applied.

The column labeled Form I applies to data systems that have the capability to use both uppercase and lowercase letters; digits; and at least the graphic characters apostrophe (') quotation mark ("), hyphen (-), period (.), and slash (/). However, they do not have the capability to use the Greek letters mu (μ) and omega (Ω).

The column labeled Form II applies to data systems that are further limited in that they have only one case (either uppercase or lowercase), and they lack the apostrophe and quotation mark.

6.4 Limitations arising from importing and exporting documents between computer programs

Despite the claim by many program vendors that their programs can "translate" into or from various formats, experience indicates that problems are frequently encountered. These problems occur even when using the same program but on different platforms. For example, Ω and μ often are not transferred properly from one program or platform to another. The transmitter of information is advised to proceed with caution and to take every step available to ensure accurate transmission. The use of proprietary programs such as mentioned above or such as T_EX or L^AT_EX may provide a solution to this problem, but the users must be on their guard. In general, unformatted text files adhering to the guidance of 6.1 are the safest editable documents to import or export.

Standard Units

Table 3—Units of the SI and units used with the SI

Unit	Symbol	Notes[a]
ampere	A	SI unit of electric current. Also, SI unit of magnetomotive force.
ampere-hour	Ah	Also A·h.
ampere per meter	A/m	SI unit of magnetic field strength.
ångström	Å	Å := 10^{-10} m Deprecated (see IEEE/ASTM SI 10-2002).
degree (plane angle)	°	
degree Celsius	°C	SI unit of Celsius temperature. The degree Celsius is a special name for the kelvin, used in expressing Celsius temperatures or temperature intervals.
degree Fahrenheit	°F	Note that the symbols for °C, °F, and °R are composed of two elements, written with no space between the ° and the letter that follows. The two elements that make the complete symbol are not to be separated.
degree Kelvin		Now called the kelvin.
degree Rankine	°R	
British thermal unit (International Table)	Btu_{IT}	Btu_{IT} = 1055.056 J
British thermal unit (thermochemical)	Btu_{th}	Btu_{th} = 1054.350 J
byte	B	A byte is a string of bits, usually eight bits long, operated on as a unit. A byte is capable of holding one character in the local character set.
calorie (International Table)	cal_{IT}	cal_{IT} := 4.1868 J Deprecated (see IEEE/ASTM SI 10-2002).
second (plane angle)	"	
minute (plane angle)	'	
foot pound-force	ft·lbf	

- The requirement is really pretty ambiguous or silent about plain-text representation of products, degrees, subscripts, and symbols

Flow from SysML to ATML

- NASA has an emerging interest in using SysML early in the design process to capture interfaces and behavior of subsystems.
- If SysML is rich enough and tools natural enough, SysML files could feed information (interfaces, requirements, design intent) to the test phase.
- Is there a strategy for natural smooth flow from SysML to Capabilities, WireLists, Common, HardwareCommon, UUTDescription, *etc*.?

Saving Aliases

- Example:
  - generic oscilloscope software saves data marked “channel 1 voltage”
  - user needs to remap this as “strain gauge 5”
- In practice most of the configurable and status variables are not central to the inquiry
- Candidate solution is an Alias Table

Saving Queries

- Expect scripts to generate queries for tables and plots of “usually relevant” variables at time of run.
- A user reanalyzing the data while writing a report or reexamining archived cold data should continue to have access to queries saved in/with the data set.

Indexing: Parametric Associations

- Performing a parametric sweep: The "observation interval"
  - Configure and settle the "stimulus" actuator
  - Reset and settle the "response" measurement
- Several ways to associate data
  - Using time-tags can be ambiguous
  - Managing an index at the "LTE" data sources is failure prone
  - Including a table associating "observation intervals" with pointers into each of the other tables is a logical housekeeping flow